DOT/FAA/AR-95/125-III, 2

Office of Aviation Research
Washington, D.C. 20591

Digital Systems Validation Handbook, Volume III

Design, Test, and Certification Issues for Complex Integrated Circuits - Chapter 2

July 1996

Final Report

U.S. Department of Transportation
Federal Aviation Administration

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof. The United States Government does not endorse products or manufacturers. Trade or manufacturer's names appear herein solely because they are considered essential to the objective of this report.

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/AR-95/125-III, 2		

4. Title and Subtitle	5. Report Date
DIGITAL SYSTEMS VALIDATION HANDBOOK VOLUME III, DESIGN, TEST, AND CERTIFICATION ISSUES FOR COMPLEX INTEGRATED CIRCUITS CHAPTER 2	July 1996
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
L. Harrison and B. Landell	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Galaxy Scientific Corporation 2500 English Creek Avenue Building 11 Egg Harbor Township, NJ 08234-5562	11. Contract or Grant No. DTFA03-89-C-00043

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
U.S. Department of Transportation Office of Aviation Research Washington, D.C. 20591	Tutorial Handbook Chapter 2
	14. Sponsoring Agency Code AAR-421

15. Supplementary Notes

Peter J. Saraceni, William J. Hughes Technical Center Program Manager, (609) 485-5577, Fax x-4005, saracenp@admin.tc.faa.gov
(Note: This tutorial is a condensed version of FAA Technical Center final report DOT/FAA/AR-95/31.)

16. Abstract

This chapter provides an overview of complex integrated circuit technology, focusing particularly upon application specific integrated circuits. This report is intended to assist FAA certification engineers in making safety assessments of new technologies. It examines complex integrated circuit technology, focusing on three fields: design, test, and certification. It provides the reader with the background and a basic understanding of the fundamentals of these fields. Also included is material on the development environment, including languages and tools.

Application specific integrated circuits are widely used in Boeing 777 fly-by-wire aircraft. Safety issues abound for these integrated circuits when they are used in safety-critical applications. Since control laws are now executed in silicon and transmitted from one integrated circuit to another, reliability issues for these integrated circuits take on a new importance. This report identifies certification risks relating to the use of complex integrated circuits in fly-by-wire applications.

17. Key Words	18. Distribution Statement
Application specific integrated circuit, Avionics, Certification, Design for test, Field programmable gate array, Fly-by-wire, Hardware description language, Reliability, Simulation, Software, State machine, Submicron design, Synthesis, Validation, Verification.	This document is available to the U.S. public through the National Technical Information Service, Springfield, Virginia 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	65	

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

ac	Alternating Current
AC	Advisory Circular
ACO	Aircraft Certification Office
AHDL	Analog Hardware Description Language
ARINC	Aeronautical Radio, Inc.
ASIC	Application Specific Integrated Circuit
ATPG	Automatic Test Pattern Generation
BICS	Built-In Current Sensors
BIST	Built-In Self-Test
BSDL	Boundary Scan Description Language
BSR	Boundary Scan Register
BST	Boundary Scan Test
CAD	Computer Aided Design
CANCER	Computer Analysis of Nonlinear Circuits, Excluding Radiation
CE	Certification Engineer
CMOS	Complimentary Metal-Oxide Semiconductor
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
dc	Direct Current
DFT	Design For Testability
ECL	Emitter-Coupled Logic
EDA	Electronic Design Automation
ESD	Electrostatic Discharge
ESDA	Electronic System Design Automation
FAA	Federal Aviation Administration
FAR	Federal Aviation Regulation
FIPS	Federal Information Processing Standard
FPGA	Field Programmable Gate Array
HCPLD	High Capacity Programmable Logic Device
HDL	Hardware Description Language
HDLC	High-Level Data Link Control
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronic Engineers
I/O	Input/Output
I_{DD}	CMOS Device dc Power Supply Current
I_{DDQ}	Quiescent State of Power Supply Current I_{DD}
kHz	Kilohertz
LFSR	Linear Feedback Shift Register
LRU	Line Replaceable Unit
MHz	Megahertz
MOS	Metal-Oxide Semiconductor
NMOS	Negative-Well MOS

NOR	Inverting Logical OR Gate
P	Power
PAL	Programmable Array Logic
PC	Personal Computer
PLD	Programmable Logic Device
PMOS	Positive-Well MOS
PROM	Programmable Read-Only Memory
RC	Resistance-Capacitance
ROM	Read-Only Memory
RTCA	Requirements and Technical Concepts for Aviation (formerly Radio Technical Commission for Aeronautics)
SAE	Engineering Society for Advancing Mobility Land Sea Air and Space (formerly Society of Automotive Engineers)
SC-180	Special Committee 180
SCR	Silicon-Controlled Rectifier
SEU	Single Event Upset
SPICE	Simulation Program with Integrated Circuit Emphasis
SRAM	Static Random Access Memory
SSI	Small-Scale Integration
TAP	Test Access Port
TMS	Test Mode Select
TTM	Time To Market
V&V	Verification and Validation
V_{DD}	CMOS Device dc Power Supply Voltage
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration

EXECUTIVE SUMMARY

If one recognizes that there is concern over software correctness issues for safety-critical systems, then attention is also warranted for complex hardware design of safety-critical systems. Digital hardware design has been on a rapid track toward massive integration at the system level, as well as at the integrated circuit level. Entire systems implemented on a single application specific integrated circuit (ASIC) will soon be commonplace.

Not only is design correctness a concern, but demonstrating that there are no silicon defects can be a major problem for developers of complex ASICs. Due to an ASIC's complexity, complete ASIC testing has become an impossible task. In addition, no single test technique can demonstrate defect-free silicon. Combinations of tests are required for uncovering different failure modes. Issues that pose threats to reliability include signal integrity, accurate timing parameters for simulation, and numerous other submicron technology-related factors.

Following are the significant conclusions from this report:

- Due to their complexity, ASICs can incorporate significant portions of a system's functionality. A silicon-based implementation may avoid the necessary scrutiny to which safety-critical systems should be subjected.

- There are no techniques and methods of design, documentation, testing, and verification identified or recognized by the FAA for today's complex hardware designs. Existing guidance does not address current practice or technology.

- There are failure modes associated with ASICs that are not readily identifiable. They can be the result of design errors or subtle phenomena that are not flagged by the tool suite, or discovered during device testing.

- ASICs incorporate design characteristics and techniques from both hardware and software disciplines. Issues relating to the hardware/software co-design process will need to be addressed in future avionics systems.

This tutorial points out that in order to generate a large ASIC, tens of thousands of lines of code can be required. Large ASIC development has been described as a "software project being performed by hardware engineers" (Corcoran 1995). Not only are there the normal hardware integrity issues for safety-critical systems, but now, all the issues of software correctness apply also. The suite of computer-based tools and hardware descriptions which are generated with a software-based hardware description language are not currently regulated by any guidelines, while tools used in software development are. Many of the issues that plague complex software systems are also emerging in complex integrated circuit (IC) designs.

1. INTRODUCTION.

While the term *complex* used in the tutorial title is a subjective assessment, the authors take the position that an integrated circuit (IC) design is complex if it is impractical to manage without the assistance of design automation tools. In general, the industry consensus is that an IC design requires the use of design automation tools when it exceeds the ten thousand gate level. At the time of this report, technology has evolved to the point where user-programmable IC designs of a million or more gates are possible.

The topic of *Complex Integrated Circuits* was identified by a number of government and industry aerospace experts as one of many areas where certification specialists require more information and training. A familiarity with a technology is essential so that valid safety assessments of systems presented for certification can be made (Janowitz 1993). Based on the rapid growth of digital technology, a number of changes have taken place in the field of digital flight control and avionic systems that call for a close examination of the risks involved with the use of application specific integrated circuit (ASIC) technology.

In the past, the design of ICs was performed by IC manufacturers. ICs were produced that were generic enough to find wide application among digital designers. In the 1980s, new technologies emerged that allowed the designers to produce small custom ICs using programmable and reprogrammable parts that were readily becoming available. Levels of on-chip integration were also increasing, allowing manufacturers to offer devices with increasingly greater amounts of functionality. Using these parts meant that IC counts could be reduced, power reduced, inventories reduced, designs changed more easily, and products produced more quickly.

The use of digital technology in aircraft is nothing new. Implementations of early digital logic ICs were relatively easy to analyze and their failure modes were well understood. While the use of ASICs in aircraft is seen as a benefit for avionics manufacturers and airframers, their implementation raises concerns about the safety of systems in which they are used. Part of the problem is due to the sheer complexity of current ASIC devices. Failure analysis guidelines that were developed for digital systems, such as SAE's ARP1834, cannot be applied in a meaningful way to the complex ICs that are being designed today. Additionally, failure modes that were non-existent with older digital technology are now prevalent and can compromise the safety of systems using these complex devices.

Commercial fly-by-wire aircraft are now being produced that have differing design philosophies from earlier aircraft. While digital avionics and flight controls have existed for a number of years, there were always backup systems that relied on different technologies, in case of a failure of the digital system. These were hybrid aircraft, using a combination of control hydraulics with interfaces to digital systems. Today's fly-by-wire aircraft, as typified by the Boeing 777, use data buses to send actuation commands to the various control surfaces, based on messages generated from the avionic systems. On these aircraft, the hydraulic link no longer exists, even for backup purposes. It is expected that if there is a failure on a primary system, another system with duplicate capabilities and connectivity will be available to take control. Back-up systems are simply duplicates of the primary systems. Redundancy may sometimes be implemented using dissimilar

hardware and software, and integrity enhanced by voting systems and other means, but the technology remains the same.

Due to increased IC densities, ASICs can now be programmed to take on tasks that were formerly performed in software. At current ASIC complexity levels, it is dangerous to assume that upset avionics are due solely to software bugs. A new urgency, therefore, exists, to ensure that these digital avionic systems are designed correctly, implemented correctly, tested fully, and are reliable in every other respect. This is no trivial matter when one considers the current complexity level of ICs. Error-free parts can no more be guaranteed than one can promise error-free software. In fact, complex ASIC design is described by Corcoran (1995) as a "software project being performed by hardware engineers," since most ASIC parts are now designed using high-level languages that can describe digital logic behavior.

Complete testing of complex ASICs is not done, since it is impractical. New testing techniques continue to be researched in order to discover new ways of enhancing an ASIC's testability. Even ICs, whose failure can cause substantial financial penalties to the manufacturer, such as the Pentium™, are not immune from process errors that can cause defective hardware.

ASICs are programmed typically by the end-user or avionics supplier. Complex ASIC designs require teams of 50 to 100 engineers. ASICs are a technology essentially unregulated by the Federal Aviation Administration (FAA) and not understood by certification engineers (CEs). The level of on-chip circuit elements that can be squeezed into an ASIC is so high that more of the software portions of avionic system designs are being placed into ASICs. ASICs are used extensively on the Boeing 777 in flight-critical systems and, along with other user-programmable logic and special function ICs, will be used almost exclusively in future fly-by-wire aircraft.

There are a number of other areas of concern relating to the use of ASICs in flight-critical systems. These are addressed in the technical report entitled "Design, Test, and Certification Issues for Complex Integrated Circuits," (DOT/FAA/AR-95/31).

2. USER-PROGRAMMABLE INTEGRATED CIRCUITS.

2.1 PROGRAMMABLE LOGIC DEVICES.

In general, a programmable logic device (PLD) is an IC that is configured by the user to perform a particular logic function, or combination of functions. Each type of PLD has a unique set of features that have advantages and disadvantages for each type of design. Cost-effective designs that can meet task requirements can be realized only if the designers account for the unique advantages and disadvantages of the various PLDs and their characteristics.

Early ICs, consisting of 14- and 16-pin small-scale integration (SSI) logic, did not allow designers any flexibility to configure their own logic. Programmable Array Logic (PAL), introduced by Monolithic Memories in 1978, is regarded as the first PLD. It used programmable read-only memory (PROM) technology to allow users to write patterns into the PAL, which configured the internal logic according to the predefined device definition provided by the manufacturer.

PLDs benefit applications where the design is expected to undergo numerous changes. Typical gate counts for these ICs run from the 100s to around 20,000. Designers make choices between PLDs based on reprogramming requirements and cost factors.

Two basic types of PLDs are in common use. One contains on-chip memory storage of interconnect patterns that are both erasable and reprogrammable. The other is based on a link or fuse technology and can be programmed only one time. Reprogrammable devices allow designers to use the same IC type for several portions of the design, as well as allowing product upgrades without having to change the printed circuit board design.

Routability can often be a limiting factor in PLDs. If the maximum number of signals that can be routed to a logic block is exceeded for a particular logic function, then the design will need to be partitioned among other free logic blocks. It can also be difficult to find an interconnect that routes all the required signals into the appropriate logic blocks. As the number of signals that are routed into a logic block approach a maximum, it becomes much more difficult to route the remaining required signals.

An additional problem occurs when there are design changes after a device has gone through a successful layout and pinout configuration. A design may no longer fit with the same pin configuration and the result may be considerable rework for the designer. Hence, routability is an important consideration in the choice of a suitable device.

Characteristics of the individual logic blocks are an important consideration in the use of PLDs. The amount of logic contained in a logic block and the flexibility of these blocks to accommodate different configurations have given designers more design possibilities. One popular logic device which was designed years ago is the 22V10. In it, the number of product terms available in a logic block is not variable. If there are terms that are not used, they end up wasting space, since they cannot be steered to another logic block, for combining with the terms of that block. Also, the logic blocks of this device do not share product terms. Therefore each logic block that requires a particular term must generate it completely. Efficient utilization of logic block resources and performance enhancement can be achieved through sharing and steering.

As improvements in the design of logic blocks progressed, vendors began to offer more options. These included steering and sharing of product terms. Product term steering allows designers to move a portion of the logic block from one cell to another. While this helps a cell that requires more functionality, the cells from which the functionality was removed are now reduced in usefulness. This lack of functionality may then result in wasted resources.

Examples of currently available PLDs include:

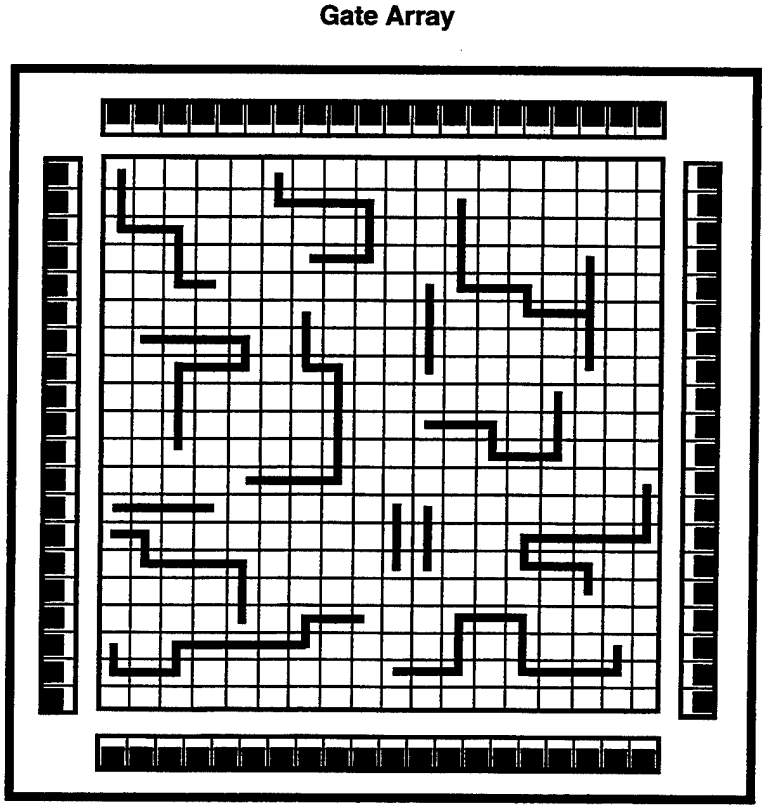
- Programmable Array Logic
- Complex Programmable Logic Device (CPLD).
- Erasable Programmable Logic Device.
- High Capacity Programmable Logic Device (HCPLD).
- State Machine Integrated Circuit.

2.2 APPLICATION SPECIFIC INTEGRATED CIRCUIT.

2.2.1 Overview.

ASICs are divided into two categories: the gate array and the standard cell. These names refer to the circuit configurations that exist within the ICs. Gate arrays can be one of two types: field programmable or masked. The field programmable version is the most complex of all PLDs. Figure 2-1 gives a simple block diagram of the gate array configuration.

The internal configuration for the gate array consists of a large array of identical logic blocks. The basic circuitry is prefabricated and requires programming the interconnects based on the design requirements. The gate array lacks the features necessary for customizing circuits, but excels in that the logic that exists can be programmed rapidly. For the gate array, the basic die is prefabricated. Only the interconnects between and in the logic cells need to be made, along with the connections to external pins.



GSC.449.95-7

FIGURE 2-1. GATE ARRAY BLOCK DIAGRAM

The standard cell configuration, as shown in figure 2-2, also is comprised of blocks of logic. However, the blocks are not all the same and they are not laid out in a uniform and repetitive pattern as are gate arrays. The basic die has not been prefabricated and layout of a complex ASIC

can be difficult. Minimization of the number and length of on-chip interconnects is a major goal in IC design and layout, in order to reduce the likelihood of layout-induced problems.

A comparison of gate array and standard cell designs is given in table 2-1.

TABLE 2-1. GATE ARRAY AND STANDARD CELL COMPARISON

Characteristic	Gate Array	Standard Cell
Nonrecurring Engineering Cost	Low	High
Per Piece Cost	High	Low
Utilization	Low	High
Turnaround Time	Fast	Slow
Customizability	Low	High

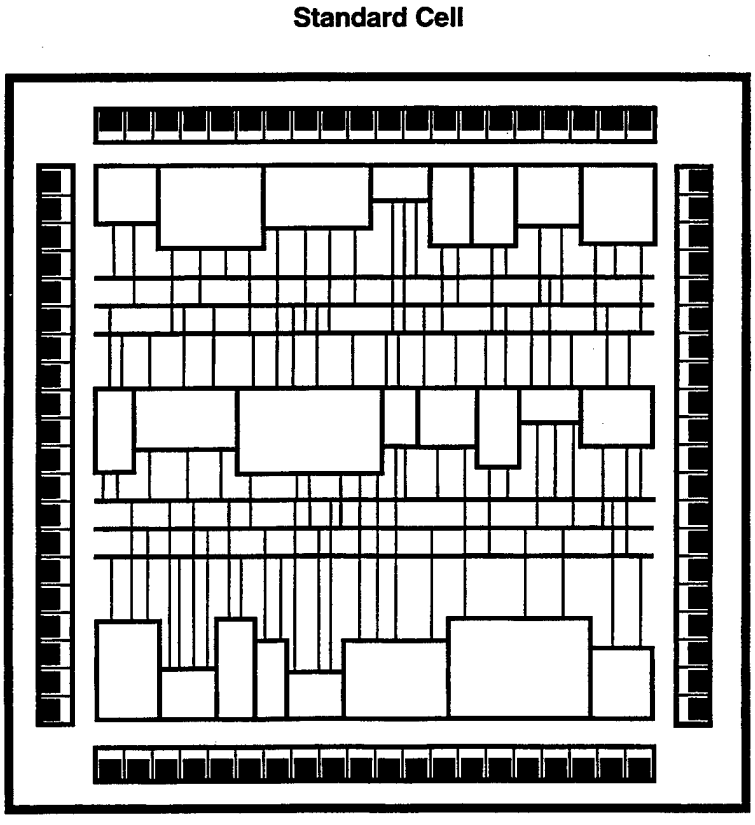


FIGURE 2-2. STANDARD CELL BLOCK DIAGRAM

As indicated in table 2-1, both the gate array and the standard cell devices have significant advantages depending on the overall application. In the gate array, where the logic blocks already exist on-chip, the nonrecurring engineering costs are lower, hence the turnaround time is faster. Also, the off-the-shelf cost of a gate array is higher than for a standard cell part. The standard cell should be used where higher device quantities are required. They have the advantage of being highly customizable since the manufacturer provides predefined libraries of common functions to the designer. Utilization of the silicon area is high given the freedom to implement only those circuit portions that are required for the design.

2.2.2 Field Programmable Gate Array.

Compared to other types of programmable logic, the gate array offers the designer more flexibility and options. Each IC is customized for a particular application by programming the internal interconnections. These interconnections consist of vertical and horizontal routing conductors that are used to connect the various internal gate array function blocks and input/output (I/O) circuitry. A cell normally consists of a register, associated logic, multiple inputs, and multiple outputs.

When designs are required that are too complex for other types of PLDs, gate arrays often provide designers with the necessary solution. Also, if large volumes of PLDs are required, migrating the design to a gate array can cut the production costs. Currently, usable gate counts for gate arrays can be in excess of 200,000.

It can be difficult and challenging to utilize large portions of the available silicon in a gate array design since there are portions of logic blocks that are not needed by the designer. There is no way to eliminate these and reclaim the space. If a more complex design is necessary than will fit into a gate array, the designer can use a more complex gate array or migrate the design to a standard cell implementation.

Increases in gate count and ease of programming these devices has led to increases in the use of field programmable gate arrays (FPGAs). Ability to design and program parts on the desktop personal computer (PC), design portability, and FPGA-specific tool support such as synthesis and simulation, allow designers to increase productivity by using FPGAs as compared to other types of PLDs.

2.2.2.1 Field Programmable Gate Array Performance.

Signal delays in the FPGA can be caused by several factors, including:

- signal wire characteristics,
- programmable elements,
- amount of cascaded logic cells, and
- propagation delay of each logic cell.

The FPGA technology influences both the interconnect delays and routability. The two most common technologies used for interconnects are static random access memory (SRAM) and

antifuse. The antifuse interconnect is smaller than the SRAM and permits many more programmable elements to fit on a die. The SRAM-based FPGA is more restricted in routability. However, SRAM reprogrammability is a considerable benefit and a characteristic that some applications may require.

Interconnect net delays for FPGAs can be significantly longer than standard cell interconnect net delays. This is due to both the signal routing architecture and the crossing points, where the signal routes connect. These points can add a significant amount of resistance to the signal networks, which in turn will cause delays in the signal propagation.

Logic cell architectures also vary for SRAM- and antifuse-based FPGAs. Antifuse FPGAs generally use a fine-grained approach to cell structure which produces smaller and simpler logic cell structures. More logic cells are required to build a function with fine-grained cells than with coarse-grained cells. A fine-grained approach, therefore, relies more on the interconnect structure and cascaded cells than does the coarse-grained approach. Since this is the case, designs that use the fine-grained approach are more likely to exhibit slower performance. SRAM-based designs must minimize interconnects due to higher delay and a smaller fuse availability (Kapusta 1995).

During the early part of the design cycle, signal delays due to logic implementation and signal routing are largely unpredictable. Synthesis tools can make estimates based on a statistical wire delay model, but the values will invariably change in the actual IC. Where high speed timing is required, the accuracies of the statistical wire delay model may not suffice.

Limitations of the synthesis tool in accurately predicting the actual timing and the layout effects on timing may lead designers to be conservative in device timing estimates. Timing-related failures may appear later in the design cycle if tight margins are chosen initially. Without holding to tight margins, however, designs become less competitive.

2.2.2.2 Advantages and Disadvantages of FPGAs.

Advantages of FPGAs include:

- FPGA use can provide greater device reliability.
- Higher densities are possible for FPGAs than with other types of PLDs.
- FPGAs allow greater architectural flexibility.
- FPGAs are more cost-effective than standard cell ASICs for lower quantities.

However, there can be some drawbacks in the use of FPGAs. These drawbacks include:

- Development cost is higher, compared to other PLDs.
- FPGAs are not as easy to use as other PLDs due to higher level of complexity.
- Hardware description language (HDL) coding may be required to manage complexity.
- Device timing is more difficult to handle for FPGAs than PLDs.

2.2.3 Standard Cell.

Designers now use ASICs routinely. Designers benefit when using ASICs for non-standard logic implementations and high volume production. Only in certain cases, such as highly cost-sensitive designs, are alternatives considered. When they are, a designer may turn to off-the-shelf or full custom components.

What distinguishes the standard cell from the gate array is that the complete mask is user-defined for the standard cell. This technology also uses full design libraries of standard logic elements and memories. ASICs can contain combinatorial and sequential logic, as well as analog circuitry. Logic forms that are implemented in the standard cell ASIC include SRAM, read-only memory (ROM), central processing units (CPUs), fuzzy logic controllers, digital signal processors, and analog and mixed signal functions.

Standard cell library elements are optimized for either high speed or high density. Clock speeds for standard cells can run to several hundred megahertz and densities are in the hundreds of thousands of gates. The design cycle time for standard cells has traditionally been longer than that for gate arrays, but new tools are minimizing that difference.

Barriers for using ASICs have largely disappeared. Tools are available to run on all common engineering platforms.

Another consideration is the breadboard. A breadboard development and test cycle would be prohibitive due to high pin counts, and manually intensive testing required. The proliferation of programmable devices, simulators, and other support tools such as in-circuit-emulators, has, to a large degree, reduced the amount of breadboard activity.

3. DESIGN ISSUES FOR APPLICATION SPECIFIC INTEGRATED CIRCUITS.

3.1 DESIGN PHILOSOPHY.

Synthesis tools are designed to create efficient logic designs. They use a high-level description of the design and produce the low level circuit design, expressed as a netlist of gates and interconnects. They run sophisticated algorithms and can tailor a design for speed or size. However, if the HDL code is cumbersome and not efficiently designed, then there is little that a synthesis tool can do to make it more efficient.

However, currently available tool sets are only beginning to address the system-level design. This aspect has become increasingly important to ASIC designers since submicron design technology has created the potential for complete systems on a single ASIC. Addressing system-level design involves techniques that apply not just to digital logic, but to software as well.

3.1.1 Top-Down Design Methods.

ASIC designs need to be performed from the top to the bottom, with knowledge of all issues from the bottom to the top. The requirements must be well-defined. Available tool sets and their capabilities must be known. Available target devices and technologies must be known.

A top-down approach will make a design easier to handle, regardless of the tools that are used. A design team is not overwhelmed by complex designs and work can be partitioned easily. A top-down approach can easily be made self-documenting. Important benefits from a top-down approach also include reduced development time and improved product quality.

An important consideration to keep in mind is that design of a complex ASIC cannot be viewed as a pure hardware task. It is, in practice, a software task that is being performed by hardware designers. Complex ASIC-based designs can contain hundreds of thousands of lines of HDL code (Corcoran 1995). ASICs are designed at increasingly higher levels of abstraction. These include graphical architectural descriptions, "C" programming language models, HDL representations, and combinations of these.

3.1.2 Other Design Approaches.

Design strategies other than top-down are used by designers when developing ASIC designs. Two other common methods are bottom-up and progressive refinement. A bottom-up method may be employed when there is a need for reusable design, with arguments being much the same as for reusable software. While it is possible to save design time with reusable parts, the potential exists for introducing problems if the reusable part does not perform precisely as the design requires.

Some designers may use a bottom-up approach when critical timing paths are necessary. Critical timing paths should be identified in the requirements and traceable down the design path. A purely bottom-up approach runs the risk of creating a design that does not reflect the design requirements.

Another approach is referred to as "progressive refinement." This is essentially a combination of top-down and bottom-up techniques. The high-level design is completed first. Portions of the design are then incrementally synthesized and tested. Incremental synthesis is performed progressively until the design is completed.

3.2 LOGIC DESIGN PITFALLS.

Design pitfalls that can affect device functionality include:

- clock-related errors,
- clock skew,
- clock gating,
- clock path length,
- asynchronous inputs and race conditions,
- asynchronous delay generation logic,
- asynchronous logic hold-time violation,

- design glitches,
- bus contention,
- asynchronous state machines, and
- metastability.

Essentially, these are design practices that should be avoided, but still manage to find their way into some IC designs. These pitfalls are based on faulty design techniques that are incorporated into digital circuit designs. A corollary with software programming practices can be made. Characteristics of poor software designs include the use of self-modifying code, excessive use of the "goto" statement, and poorly documented code. Further information on these pitfalls can be found in the technical report titled "Design, Test and Certification Issues for Complex Integrated Circuits," DOT/FAA/AR-95/31.

The use of tools and higher levels of design abstraction, the pressures of rapid time to market (TTM), inexperienced designers, and other factors often contribute to designs of inferior quality. With the availability of more powerful design tools, the actual logic implementations are further removed from the designer's critical inspection. Designs often use libraries of functions which are supplied or purchased along with the tools. Both the tools and design libraries may contain design flaws that can escape the notice of designers. Higher levels of abstraction mean that those who are not as familiar with digital design techniques and practices can now perform design functions. What can suffer is the ability of the designer to verify that the circuit implemented by the tool suite is correct. Those tasked with design verification should ensure that design pitfalls are avoided and that good design techniques are applied consistently.

3.3 ISSUES IN SUBMICRON TECHNOLOGY.

Shrinking IC geometries is key to several benefits for both IC manufacturer and end user. The number of transistors can be increased, yielding greater functionality per unit area. Performance is increased, since smaller geometries allow higher clock rates. Also, the ability to put greater functionality into a smaller area can yield an overall power reduction.

Accurate estimates of design parameters such as power dissipation and delays take on greater significance as designs move into the submicron and deep-submicron (0.5 micron and below) regions. Unless a strong link exists between front-end and back-end tools, accurate estimates will not be possible. Front-end tools handle tasks such as synthesis and partitioning, while back-end tools handle layout and block placement issues. Without a good correspondence between these tool types, the designer of a complex ASIC is destined to perform a higher number of iterations before reaching an acceptable design.

Some of the key issues that designers will confront when dealing with submicron designs include:

- Accurately accounting for interconnect delay as the dominant delay factor when modeling the ASIC and design analysis based on high frequency operation of the IC.

- Designing with an accurate estimate of the power dissipation, both for potential "hot spots" and for the entire IC.

- Managing the stronger electric fields as transistor line widths decrease.

- Integrating IC package parameters into the simulation.

- Managing a complex IC design database that can exceed 1 gigabyte for current designs.

Two major objectives of ASIC designers are to implement the desired functionality and to do that within the given constraints. A device specification will include parameters such as power consumption, die size, packaging, and noise margins. HDLs are generally utilized to implement ASIC functionality. The gate-level netlist generated from the synthesis process determines the major portion of the physical design characteristics. However, unless these characteristics are strongly linked to some of the key design parameters for complex ICs, the design may be plagued by a number of potentially costly problems requiring more design iterations.

Submicron design issues relating to gate delay, floorplanning, crosstalk, power, and thermal design are found in the technical report DOT/FAA/AR-95/31 mentioned earlier.

3.4 NOISE AND GROUND BOUNCE.

Signal integrity within an ASIC is a factor of which CEs need to be aware. Predicting signal integrity can be a difficult task for designers. Complicating the matter is the fact that there is a lack of tools that take signal integrity issues into account during the IC layout phase. Often the tools, technology, and methods are not identified sufficiently early in the total process. The device packaging technology is often put off until the final design stages, while the semiconductor technology is normally chosen during the early design stages. When this is done, two problems can result: inability to predict performance and inability to meet performance expectations.

Noise issues are essentially the same for IC designers as for printed circuit board designers. However, the problem complexity increases along with IC complexity. Since signals are closer together with each technology advancement, RC-based effects increase due to closer signal wires and the increasing resistance of smaller signal wires. One of the complicating factors is that those who are skilled in digital technology are not skilled in analog techniques. As device complexities and clock rates increase, signal integrity becomes more of an issue.

3.4.1 On-Chip Propagation Characteristics.

Low voltage operation, high complexity, and high speed are favored by miniaturization. However, two major drawbacks that must be overcome by miniaturization are high values of wire resistance and thin dielectric fabrication. Both are critical issues for miniaturization, since both can impact signal integrity severely.

ASICs and memory ICs are fabricated using multiple layers of metal for interconnections between the various circuit elements. Memory ICs have regular patterns and require from one to three layers

of metalization. Complex ASICs (which may also contain memory) generally have much more random patterns and can require many more layers of metalization in order to implement the interconnection of circuit elements. While the technology exists to fabricate many levels of metalization, there are other issues that need to be addressed during the design, such as signal integrity, signal synchronization, and fabrication cost.

Another difference between old and new IC technology is that on-chip signal propagation loss, which once was not a factor, must now be accounted for in complex designs. Formerly, signal propagation did not need to account for line propagation delay, since lines were physically larger (lower resistance) and gate delays were also much larger. Delay is now RC-dominated. Since line geometries have become small in order to support the massive amounts of on-chip circuit elements, line resistance has increased enough to be the dominating delay factor for on-chip signals.

Ideally, low loss lines are required that do not take up much of the available area. However, low loss lines are difficult to implement, since, as signal speeds increase, "skin effect" also increases. Skin effect causes a high speed signal to travel near the surfaces of the conductor, without taking advantage of the entire cross section of the conductor. As a result, the ac resistance will be higher in value than the dc line resistance. In order for the designer to be able to model critical paths accurately, it is essential that the propagation properties be modeled accurately.

When simulation is used to model on-chip interconnections in a complex IC with small geometries, a number of factors that influence signal propagation must be taken into account. CEs need to be aware of device geometries to ascertain if the modeling accounted for all contributing factors.

3.4.2 Consequences of Noise.

Timing problems due to signal skew can cause failure of the IC. This failure can take the form of a hard logic fault and cause problems such as bus contention on read and write cycles. Logic faults generally are rare in designs that are simple. Other observations that can be made about on-chip logic faults include:

- Faults are rare from crosstalk alone, and are usually the result of a combination of factors.
- Faults are common in systems that have a large number of simultaneous switching events.
- Faults are common where inaccurate timing estimates are made.
- Faults generally are nonlinear and data dependent.

While timing-related faults are rare in simpler circuits, they increase in probability if all factors contributing to timing delays are not characterized well, or if they are not characterized for all combinations of logic states. Since signal edges are the measuring rod of circuit timing, anything that influences these edges must be taken into account. Signal edges are affected by crosstalk and signal reflections.

Noise is generated when there is an incremental demand for current from a switching device, such as a transistor, and is referred to as ΔI noise. When multiple devices switch simultaneously, the

incremental demand multiplies. Inductance in signal paths generates L(di/dt) voltage. ΔI noise can also affect signal edges, even if the noise is not severe.

Contributions to signal edge degradation can be insidious, since they are data dependent. These contributions can vary based on the state of the logic or the polarity of the logic state change. In propagation where delay is characterized by RC effects, how the capacitances are modeled can influence the accuracy of timing predictions.

Without taking these characteristics into account, tools will not be able to produce accurate timing models. Tools that can be affected include ones that model clock distribution, timing synchronization, and propagation delays. Accurate modeling of these parameters becomes increasingly important as device complexity increases.

3.4.3 Recommendations for On-Chip Design.

High speed and signal quality can suffer when the design of the current return path receives insufficient attention from IC designers. Some of the problems that are experienced with current return path design include:

- path resistance is too high,
- return path is too long,
- return path exhibits semiconductive behavior, and
- return path is shared by too many interconnections and other circuits.

Many other concerns need to be addressed by designers, especially since IC technology has progressed to below the submicron level. Unless noise truly originates from an off-board source, it is a design problem. Noise can originate from within the IC or can propagate from printed circuit board traces, to the IC pins, and into the IC. Being data-sensitive, it may not be recognized as a problem until devices suffer field failures. It is more a problem in complex and high speed ICs, such as ASICs. CEs need to be aware of these issues and check to see how manufacturers have addressed them.

3.5 LATCH-UP.

Fabrication of Complimentary Metal-Oxide Semiconductor (CMOS) ICs involves silicon processes that create parasitic silicon-controlled rectifier (SCR) circuit structures. While not intended to operate in the SCR mode, these structures, when subjected to certain conditions, exhibit SCR characteristics.

As is the case with an SCR circuit, once the current starts flowing, it cannot be stopped without interrupting the emitter-collector current path. Hence, even though the source that originally caused the current flow to begin may have been removed, the current flow continues. In the SCR configuration, the transistors operate as low-resistance, high current switches. Once this occurs, this latch-up current flow can cause permanent damage to this circuit, to the transistor junctions, or to the metal lines that connect the circuit elements.

I/O pins can be very noisy as a result of being connected to printed circuit board traces that are subject to capacitively coupled signals from other traces. Overshoot of a signal can occur when a fast switching signal is driving a capacitive load such as the printed circuit board trace. A transient forward bias condition at the I/O transistor junction can result. Latch-up is likely to be induced under these circumstances. The occurrence of latch-up can be reduced using care in the design and layout of the printed circuit board and attached devices.

Latch-up is also viewed as a failure mechanism in IC reliability research. It can result from improper design or fabrication techniques. Quality control in the fabrication process is essential for the reduction and elimination of latch-up problems.

3.6 SINGLE EVENT UPSET OF DIGITAL LOGIC.

The upset of digital systems has long been recognized as a possibility for satellite and other space-based systems. Older IC technologies are unlikely to be influenced by single event upset (SEU). Until recently, it was not considered a problem for commercial avionics. However, as submicron technology advances continue, experts believe that SEU will become more and more a problem. According to Keller (1993):

Systems integrators, avionics manufacturers and even some of the major commercial aircraft manufacturers are finding that cosmic radiation poses an SEU threat to avionics flying at altitudes exceeding 10,000 to 15,000 ft.

The most vulnerable devices as far as susceptibility to SEU are SRAMs. This is due to the way SRAMs are fabricated. The individual transistors that make up the memory cells are tightly packed. Each new generation of memories further reduces the spacing between transistors, increasing the probability of a particle hit. As memory dimensions continue to decrease, it is possible that upsets could occur as often as once per flight, unless precautions are taken.

In a study conducted by Boeing, it was found that SEU rates increased by a factor of 2.2 between mid and high altitudes, and that an additional increase of the SEU rate by 2.1 occurred when going from mid to high latitude. It was observed that the SEU rate over Norway is close to those found in low earth orbit (Keller 1993).

Protection from SEU is a design issue. Memories that may be incorporated into ASICs and FPGAs can be upset by cosmic radiation. Additionally, other logic can also be upset. Radiation can induce transient upset into combinatorial circuits. If the upset persists long enough to meet certain minimum register requirements, such as pulse height, width, and timing, the error becomes "switched in" to produce a stable bit error (Baze et al. 1993).

3.7 CONSIDERATIONS FOR PRINTED CIRCUIT BOARDS.

Signal integrity is a problem not only on-chip, but off-chip as well. One of the reasons why noise is such a problem is that there are impedance discontinuities along the path of a signal, as a signal travels from one IC to another. These discontinuities cause signal reflections that result in ringing and crosstalk. Noise that originates from within the IC is easily conducted off the IC. Noise

generated from outside the IC can be conducted through the traces, socket, and device pins into the IC.

Crosstalk can also be an issue of concern for printed circuit board designers. The same effects that occur on-chip are also present off-chip. Long parallel traces and fast switching signals are two of the main contributors to printed circuit board crosstalk.

3.8 PERFORMANCE MEASUREMENT.

Device performance is often unknown until the ASIC is programmed and exercised in the target system. Accurate estimates on performance can be difficult to make. With FPGAs, for instance, it is necessary to "implement the design before knowing what the final performance will be" (Kapusta 1995). The placement and routing of a design determine the propagation delay and maximum operating frequency. Timing predictions for FPGAs are therefore very difficult and need to be finalized in the implementation.

Generally, timing is easier to predict when the devices are smaller. PLD timing prediction relies upon the manufacturer's data book specification. Timing for CPLDs is generally more difficult to predict than other smaller PLDs. CPLD timing can be dependent upon factors such as the number of product terms or the fan-out. The implementation decisions are typically done by a software logic compiler. Therefore the performance may be unknown until actual device testing can be performed. Some CPLDs have simpler timing specifications, allowing the designer to predict accurately beforehand the expected performance.

4. TOOLS AND TECHNIQUES FOR HARDWARE INTEGRATED CIRCUIT DESIGN.

4.1 EARLY DESIGN TOOLS.

The first significant tools that became available to assist IC design were computer aided design (CAD) tools. CAD tools were designed to assist in the layout of printed circuit boards. The printed circuit board was a target for greater IC integration by reducing the trace widths, spacing, via holes, and distance between the ICs.

As digital technology continued to evolve, other computer-based design tools emerged. These included schematic capture, netlist generators, and fan-out checking tools. When PLDs became available, new design methods became essential. While manually designating dots for each interconnect of a PLD is sufficient for smaller PLDs, it becomes tedious and error-prone for more complex PLDs. The current PLD complexity level is too great for traditional techniques. Other factors such as development cost and TTM are also driving the requirement for tools that automate IC design.

4.2 APPLICATION SPECIFIC INTEGRATED CIRCUIT TOOLS.

Many of the advances in the digital circuit design field are related to the needs of ASIC designers. Logic synthesis tools and testing tools have several generations of improvement as the *average* size of design implementations for ASICs has increased to over 50,000 gates. Design platforms on

which ASICs and other user-programmable device design tools run have evolved from the minicomputer and mainframe, to the commonplace workstation. Tools are also now commonly found running on the high-end PCs which rival the performance of low-to-midrange workstations.

Along with the need to keep tools in step with increasingly complex very large scale integration (VLSI) technology, came changes in the way designs were captured and translated into a digital format. One of the essential steps in the technology progression was to move to higher levels of abstraction in the representation and design of digital circuits. This brought about a new category in digital design representation: behavioral models. Instead of drawing the actual logic gates, registers, counter, and other circuits, the behavior of those circuits is described in an HDL. The actual implementation is then left to the tool suite. Once a particular language is chosen, a logic synthesis tool is used to take the high-level circuit description and translate it into a netlist. This netlist describes the circuit elements and their connectivity. The netlist contains the same information that would have been produced, or deduced, manually from a schematic drawing.

The synthesis tool takes as input some form of HDL. Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) and Verilog HDL are currently the two most popular HDLs, although there are a number of others. Being the two most used HDLs, they also benefit by having the greatest amount of support from tool vendors who create tools to be compatible with these HDLs.

With the rapidly increasing gate densities the designer's productivity is struggling to maintain pace. Electronic design automation (EDA) vendors have responded to this problem by offering a number of tools that are aimed at solving a small portion of the ASIC design task efficiently. According to Beechler and Hanz, (1994):

Today the gate array design tool suite is composed of a "best of breed" approach, in which companies piece together the best "point tools" for each phase of the design process.

Associated cost for ASIC hardware development includes the tool suite, computer hardware, training, tool familiarity, and CAD support personnel. Often, developers purchase, create, and reuse libraries of HDL-based designs. When the design is finished, the netlist for the ASIC is sent to an ASIC vendor (foundry) for placement and routing. The vendor fabricates the prototypes and returns them to the developer for testing.

User-programmable ICs, such as PALs and HCPLDs can relieve some of the workload, time, and expense related to the ASIC development cycle. Where the placement and routing phase is typically done at the foundry for ASICs, it is done at the designer's office for PALs and HCPLDs. A computer file is then used to program the IC using one of the numerous device programming tools available. Rapid iterations are possible since these devices are user-programmable. Simple changes can be made and new devices produced in a matter of minutes.

With the pressure building steadily to improve software tools, silicon suppliers are strongly emphasizing open systems that make best use of third party EDA houses. Some vendors allow designers to mix and match tools from third party vendors. Design costs for ASICs are growing

along with the complexity. Design houses need to maintain a closer relationship with the customer in order to maintain the expected fast turnaround times (Waller 1995).

4.3 DESCRIBING THE DESIGN.

While dramatic improvements have been made in tool capability, describing a complex ASIC is a time consuming task that can require a large team of designers. Design tools have not evolved sufficiently to produce designs based on a thorough specification. Detailed design descriptions must be facilitated by some other means. Common methods that designers use for describing logic designs include:

- hand drawn schematic,
- state machine,
- waveform,
- logic description tools such as PALASM™, CUPL™, and ABEL™, and
- HDLs such as VHDL, Verilog HDL, and others.

Some of these techniques are described in more detail in DOT/FAA/AR-95/31.

4.3.1 Very High Speed Integrated Circuit Hardware Description Language.

The Department of Defense mandated the use of VHDL in 1987. It became a government standard when it was made part of the Federal Information Processing Standard (FIPS) Publication 172. Computer and communication systems are designed to these standards, and as of January 1993, all digital systems supplied to the government are required to be produced in VHDL. Adoption of this standard is intended to reduce production times and life cycle costs for digital systems procured by the government.

The F-22 Advanced Tactical Fighter is the first platform designed under a United States Air Force mandate for all systems to use VHDL for top-down design. In this, and other complex systems, it is necessary for many contractors to share design data. Errors encountered by one contractor interpreting a specification differently from another can severely impact project cost and schedule. Modeling complex systems, such as exist on the F-22, in an HDL that can express design intent clearly, makes design changes easier to distribute and can make complex designs easier to manage.

Schematic entry methods are more cumbersome as gate counts increase. HDLs have made the designer's task easier for the following reasons:

- Changes are more easily made on a computer than on paper.
- HDLs isolate the designer from constantly changing technology.
- Designs can be expressed architecturally and behaviorally allowing the HDL synthesis tool to complete the design.
- HDLs allow for hardware reuse and extensive use of libraries.

- HDLs allow for standardization among different vendors.

There are also drawbacks for VHDL.

- While VHDL does allow for standardization, it currently is hindered by a lack of uniformity among various vendors, such as in features that may or may not be implemented in the vendor's particular version of VHDL.

- The different subsets of VHDL used by various vendors hinder the language's portability and ease of design migration.

- VHDL handles gate-level complexity. It does not handle gate-level timing. It is easy to make mistakes with signals arriving late from other VHDL entities.

4.3.2 Verilog Hardware Description Language.

Verilog HDL is the most widely used HDL. Verilog HDL started as an input language for the Verilog logic simulator. In 1991 the language was put into the public domain and many vendors now provide synthesis and simulation tools based on Verilog HDL.

Verilog HDL is a structured language that has capabilities similar to VHDL. Vendors support Verilog HDL with libraries and tools that accept it as input for simulation and synthesis. Although it is not standardized as is VHDL, there is an Institute of Electrical and Electronic Engineers (IEEE) committee that has been working on adopting Verilog HDL as a standard.

Many designers hold that Verilog HDL is easier to learn and use than VHDL. Arguments over which HDL is better have been ongoing for years. Essentially, the advantages and disadvantages for VHDL also apply to Verilog HDL.

4.3.3 Analog Hardware Description Languages.

While HDLs for digital systems are commonplace and some have been standardized for a number of years, at the time of this publication, there is no official analog hardware description language (AHDL) standard. There is an IEEE standards committee working on an analog extension standard, VHDL 1076.1. This extension is commonly known as VHDL-A. Also, some interest exists in proposing an analog extension to Verilog (Verilog-A) for possible adoption as an IEEE standard.

System-level analog design and simulation would benefit from the availability of an analog HDL standard in the same way that HDLs for digital systems benefit designers. The availability of a library of behavioral analog models that are fast, accurate, and robust is required. Analog synthesis, which is a driving force for the development of AHDL, is still in the early stages of development.

Simulation was one of the early needs for the analog designer. Computer Analysis of Nonlinear Circuits, Excluding Radiation (CANCER) was one of the original nodal analysis programs, developed in 1969-1970. This program evolved into Simulation Program with Integrated Circuit

Emphasis (SPICE), then into SPICE2. SPICE, along with its derivative programs, is likely the most widely used circuit simulator.

There are problems facing AHDL development that make it more difficult than HDL development. The analog language must be able to describe noise, complex statistics, and second and third order effects, as well as other unique analog characteristics. Accounting for these complex effects is necessary to make the AHDL design useful. Practical top-down analog design requires an accuracy of five to 20 percent of the SPICE solution (Coston 1994).

4.4 SYNTHESIS OF DIGITAL LOGIC.

A great amount of research has taken place and is still ongoing in the area of logic synthesis. While a number of synthesis tools are now available, further research is needed in specific areas. Synthesis tools are a necessity for dealing with the complexity of modern digital design. Without automation tool support, ASIC design today would be quite impractical. Synthesis tools have increased design efficiency significantly. Designers are able to spend more time describing the design and setting various parameters and constraints using synthesis tool scripts, allowing the tool to handle the more tedious portion of the design.

While design efficiency is increasing, device line widths are decreasing, and geometries and metal layers are increasing. This leads to yet higher levels of device complexity with no end in sight. Tools often are in the "catch-up" mode, allowing designers simply to cope with this complex technology, but not to manage it efficiently, as in the case of a mature tool suite. High-level synthesis techniques seek to address this problem.

When considering the number of gates that can be designed into an ASIC, the schematic design entry method has little practical value for most current designs. This method is a disadvantage for larger designs. Only where the number of gates is small can a schematic-based design be considered practical. While schematics or netlists are still a part of the design process, they are created by the synthesis tool after the gate-level design is created.

A schematic capture-and-simulate design methodology had been used extensively until recently. Often the design requirements are received by the designers with no guidance concerning the implementation. A block diagram may then be drawn, which serves as a preliminary specification. This block diagram may be refined further before it is passed on to a team of logic designers.

The designers convert each of the blocks into a logic schematic. This schematic is then captured using a schematic-capture tool. A simulation is run to verify timing, functionality, and fault coverage. Fault coverage refers to the capability to control and observe internal circuits of an IC. The captured schematic is then used as input for placement and routing tools.

4.5 SYNTHESIS OVERVIEW.

Synthesis involves the incorporation of three processes into a single tool. These processes are HDL translation, mapping, and optimization. The translation process takes the HDL and converts it into boolean equations. Mapping is a process of synthesis whereby the defining design equations are

translated into a specific technology-dependent component library, or gate design. Optimization is performed in order to create a product that favors a particular characteristic, such as low power consumption, testability, area reduction, or high speed operation.

Synthesis tools should be sufficiently flexible to allow the designer to specify boundary conditions for an optimal design. Some of the parameters and constraints that designers need to specify include:

- signal drive capability;
- rise times and fall times for various signal paths;
- maximum fan-out;
- operating conditions, such as voltage, temperature, and packaging;
- models for wire loading;
- clock data such as frequency, and setup and hold-times; and
- design hierarchy preferences.

The synthesis tool performs a number of steps in the process of arriving at a gate-level design. Figure 2-3 shows these steps and the order of execution. At the top level is the HDL. For large designs, this is typically implemented with Verilog HDL or VHDL.

An HDL compiler and parser is the next step in synthesis. At this step the HDL syntax is checked. A translation of the HDL description is made into high-level equation form. A high-level structure is created for use in the following steps.

Partitioning and hierarchical flattening is performed in the next synthesis step. Partitioning breaks up the design into parts that should be synthesized separately. This is done to make the design more manageable and more easily synthesized. A good synthesis tool can greatly reduce the number of interconnects necessary between different parts of the design.

Hierarchical flattening consists in merging together levels of design hierarchy in order to simplify a portion of the design and obtain a single level. Boolean flattening removes intermediate variables, minimizing the number of logic levels. Gate count is reduced and processing speeds are increased with boolean flattening due to fewer stages between inputs and outputs. Examples of hierarchical and boolean flattening are shown in figure 2-4(a) and (b).

Mapping is a portion of the synthesis process that relates the boolean representation to a particular component library. Component libraries are technology dependent and supplied by the ASIC vendor. As with software libraries, ASIC libraries are developed from HDL descriptions and then compiled into a binary library file. Actual library designs differ based on whether a design is based on CMOS, emitter-coupled logic (ECL), or other implementation technologies.

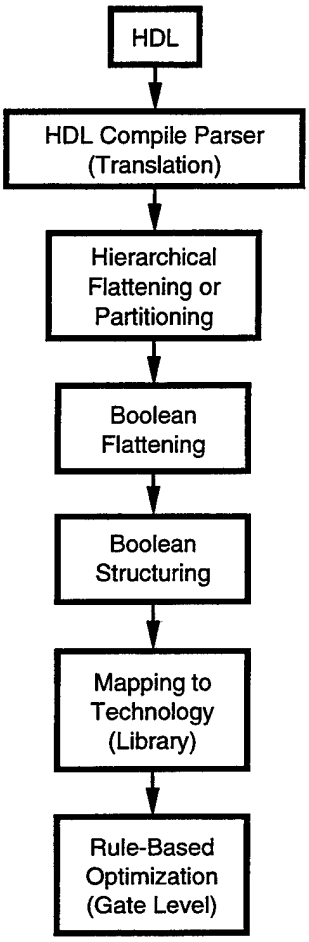


FIGURE 2-3. LOGIC SYNTHESIS PROCESS
(Widman 1994)

Along with a functional description, libraries contain design details such as:

- setup and hold requirements,
- wire loading data and fan-out limitations,
- signal rise and fall times, and
- area.

Optimization occurs as the final step. This is a gate level optimization that is designed either to reduce the area used by a design or to increase the speed of a design.

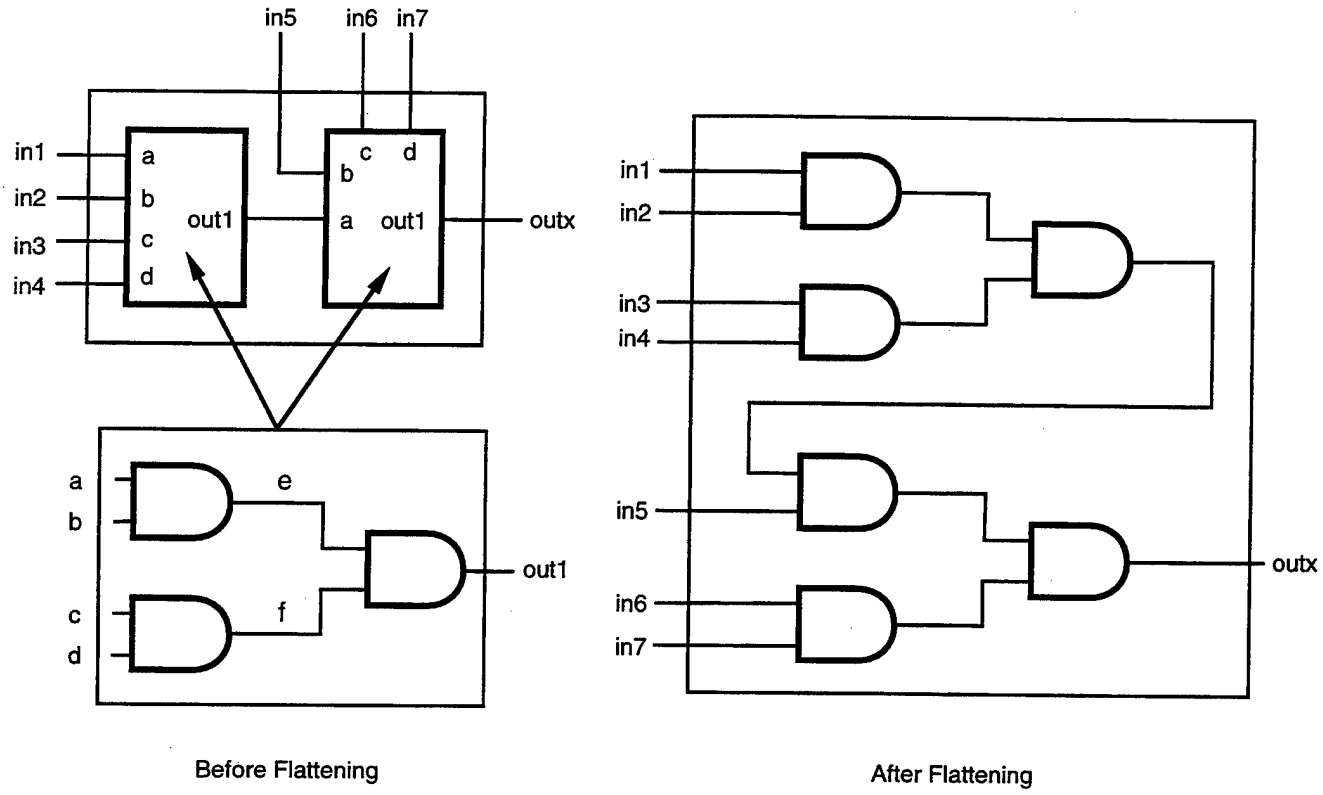


FIGURE 2-4(a). HIERARCHICAL FLATTENING

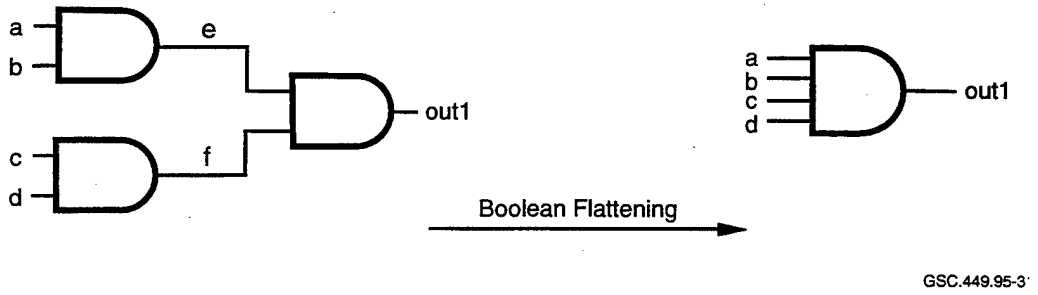


FIGURE 2-4(b). BOOLEAN FLATTENING

Prior to synthesis, designs were performed manually. The engineer worked from a specification to create a gate level implementation. The end result is a schematic drawing which depicts the elements of logic comprising the design. When design is performed this way, optimization is both difficult and time consuming. Also, this method is tied closely to a specific technology. Changing to another technology can be time consuming.

Switching from a schematic-based to a synthesis-based design involves several methodology changes. Some old tools are no longer needed, while other newer tools must be learned. A gate change is easier to do using a schematic. An equivalent change made by an HDL and then synthesized may produce other changes, since the correspondence between the HDL and netlist produced as a result of synthesis can be obscure.

There is no direct correspondence between the number of statements of HDL code and the number of gates that are generated by the synthesis tool. A high-level description can generate few or many gates. Small HDL program loops can cause a large number of gates to be created. Detailed and large descriptions are sometimes reduced to a small number of gates. However, if the HDL code is broken down into groups whose code-to-gate correspondence is known, then reasonable estimates of gate counts can be achieved.

4.6 ELECTRONIC DESIGN AUTOMATION TOOL STANDARDS.

One of the biggest problems faced by the EDA industry is interoperability among tools from different vendors. While vendors offer large tool suites for ASIC design, no single vendor provides designers with the complete set of tools that is necessary to take a design from start to finish. Thus designers are forced to use tools from multiple vendors. Porting designs from one vendor's tool to another can be a time consuming, expensive, error prone, and possibly unsuccessful venture.

At the core of the interoperability problem is the data exchange problem. Design data can have complex and varied representation, depending on its use. Data are used in representing all phases of design such as schematic capture, simulation, placement, and routing. Different vendors also represent data for similar functions in different forms. For schematics, one simple example can illustrate the point. For some, an intersection of two lines on a schematic means that there is an electrical connection at that point. For others, this intersection is simply an open circuit, unless there is a connecting dot at the intersection.

Overcoming interoperability problems can be difficult. In order to do so, in-depth knowledge of the internal data models for the various tools is required. Many tool developers view these data models as proprietary. Hence, efforts aimed at standardization can face considerable difficulty.

Most tool vendors claim commitment to various standards. In reality, tools generally provide the interface for reading in various standard data formats while not providing any choice for data-out formats. Vendors typically do not provide a data-out interface since it is a competitive disadvantage and allows the user to switch to another vendor's tool. Standardization issues will continue to plague the industry unless serious standardization efforts are undertaken by all parties (Donlin 1995).

4.7 FUTURE DESIGN TRENDS FOR HARDWARE.

Significant change is underway in the design automation field relating to how design is performed. One area where fundamental changes are starting to emerge is in design specification. Instead of using an HDL, researchers and manufacturers are seeking to raise the level of abstraction to a graphical representation that isolates the designer from the HDL.

In the future, design automation tools will be used to codesign systems, generating both HDL code and software. System behavior is captured in statecharts and analyzed using an executable specification. This method has already been applied successfully to real applications with results that demonstrate marked improvements in the development cycle.

These tools focus on assisting designers to manage design complexity. Since digital system complexity continues to increase, automation tools will play even larger roles in the future than they do currently. CEs need to be aware of trends that will eventually play a large part in the design of avionic systems. A unified approach in the certification of systems produced by these tools, that will consider the relevant safety issues, from both hardware and software perspectives, will be required.

5. FABRICATION AND RELIABILITY ISSUES FOR THE PHYSICAL HARDWARE.

Traditionally, reliability issues for digital ICs focused on physical failure mechanisms. However, for complex ASICs, the term "reliability" has taken on new meaning. Physical failure mechanisms are not the only source of device failure and new techniques for reliability calculations are needed. It cannot be assumed that errors in the design or implementation phase will all be detected and corrected before devices find their way into products.

5.1 RELIABILITY AND SOFTWARE ISSUES.

Currently, complex ASICs are being developed that require hundreds of thousands of lines of high-level code to describe. According to Munson and Ravenel (1993), the pattern of software faults has been shown to be distinctly related to the complexity of the software. The physical barriers that would limit the complexity growth of ICs have not yet been reached. New technologies and techniques are continually under development. The level of required coding is so high that new techniques are being researched and developed to manage the code complexity problem.

Hardware designers, who often are not trained in programming techniques that lead to high quality code, write the HDL descriptions. On the other hand, some ASIC design is being done by software programmers who generally understand modern programming techniques. Even if this could allay some of the fears relating to the use of proper programming techniques, there remains a larger problem of simulation and verification. A person unfamiliar with hardware cannot address or comprehend what is necessary to perform and interpret these functions.

5.2 RELIABILITY AND HARDWARE ISSUES.

IC reliability continues to be an issue of great concern. Increased complexity means higher pin counts. One of the limiting factors for ASIC designs is the number of available I/O pins. Some devices contain over 500 pins and pin counts close to 1,000 will not be uncommon in the near future. There are many potential problem areas just related to packaging.

Since complete testing of complex ICs is not currently practical, device failures under typical operating conditions can go unnoticed. Furthermore, how can it be demonstrated that no failures would occur during "shake and bake" testing if complete testing is not performed? Due to the

nature of certain failure modes, stressing environmental parameters may be the only way to induce a particular failure. For instance, a complex state machine controller may have a large fan-out for the clock signal. Suppose that due to thermal cycling, the clock timing changes sufficiently to introduce an operational error in the sequence of the state machine. That error can go unnoticed in test, unless the particular test for that condition is executed during thermal cycling. While this may seem trivial for designs containing small state machines, it becomes far more complex when one considers that state machines with thousands of states can be designed and implemented easily in the modern design environment.

Reliability is commonly defined as the probability that a device will operate correctly, for a specified amount of time, in a specified environment. Proper design techniques can assist in eliminating some failure mechanisms. Electrostatic discharge (ESD), for example, can cause reliability problems. Following adequate design rules can reduce or eliminate reliability problems due to ESD.

5.2.1 Failure Mechanisms.

Failure mechanisms for ICs exist in two categories: process anomalies and wear-out mechanisms. Process anomalies are caused by problems in the manufacturing process, resulting in defects such as contamination and ESD damage. Process anomalies are normally detected by "quality" procedures such as visual inspection, burn-in, and thermal cycling. If anomalies escape detection of the quality screening, they can cause acceleration of wear-out mechanisms. Undetected ESD damage, which can weaken thin metalization or insulating oxides, may result in dielectric breakdown or electromigration failures earlier than normal in the life cycle.

Wear-out mechanisms result from certain intrinsic properties of materials. Hence, a reduction in wear-out-related failures requires design-level measures, tight process control, and high quality materials. There are three primary failure mechanisms that receive much of the attention from reliability engineers. They are latch-up, electromigration, and time-dependent dielectric breakdown.

5.2.2 Accelerated Testing.

In order to reduce the number of early life failures in basic CMOS devices, testing of physical failure mechanisms is performed. This involves accelerated life tests and other stress tests. Typically, a manufacturer will select devices from multiple wafer lots to ensure that no process variations occurred. A number of tests are then run to ensure that all devices meet some selected quality standard. Examples of tests that are performed include high-voltage operating life, high-temperature/humidity/bias, temperature cycle, and thermal shock.

5.3 RELIABILITY AND THERMAL CONSIDERATIONS.

Increased silicon area and higher clock rates have also contributed to the rising difficulties of thermal management. From the design side, every effort should be made to minimize power consumption. Accurate estimates of the final device power dissipation are needed before the device is cast in silicon. Thermal management has become more difficult as submicron device technology

has generated multi-million transistor devices. An IC's junction temperature is a key variable in the equation of the device's long-term reliability. Long-term reliability of semiconductors degrades proportionally with temperature.

6. APPLICATION SPECIFIC INTEGRATED CIRCUIT TEST CONSIDERATIONS.

6.1 UNDERSTANDING TEST.

As technological advances reduce the physical size of the CMOS transistor and improve silicon quality, applications demand more functionality. The result is that more transistors are packed in an ASIC device. Complexity is the primary reason why ASIC testing has not been more thorough. The testing methods most critical to an ASIC's success and quality are the behavioral and non-behavioral functional tests. Their goal is to ensure no device faults exist that can cause an ASIC to malfunction in operation (Levitt 1992).

Behavioral testing exercises the ASIC device in the same manner as it is operated in the system. Any limitation in test capability could mask a problem and thereby allow defective devices to be placed in operation. This could be disastrous for a safety-critical application, such as an aircraft flight control system. Non-behavioral testing, referred to as structural testing, checks to see that each circuit and interconnection is operational and performs its individually designed function.

As the functional complexity of ASICs and the resulting gate counts continued to increase, manufacturing test became unmanageable. A design for testability (DFT) approach was developed by the industry as a solution to the ASIC testing problem. The intent of DFT strategy is to minimize the number of faulty ASIC devices by achieving the highest level of fault detection possible (Gruebel 1995). If the fault detection program produced a fault coverage of 100 percent, 100 percent of the defective devices would be detected in test as shown in table 2-2. There would be zero undetectable defective devices.

Table 2-2 parameters are defined as follows:

- Percent of Undetectable Defective Devices is the number of defective devices that tested good divided by the total number of devices that tested good, expressed as a percentage.

- Device Fault Coverage % is the number of detectable device faults divided by the total number of possible device faults, expressed as a percentage.

- Process Yield % is the number of devices that test good divided by the total number of devices tested, expressed as a percentage.

Table 2-2 shows, for a device fault coverage of 90 percent and a process yield of 90 percent, 1.05 percent of the devices could actually be defective. When the device fault coverage is 100 percent, zero undetectable defective devices will be produced regardless of the process yield value.

TABLE 2-2. PERCENT OF UNDETECTABLE DEFECTIVE DEVICES
(Gruebel 1995)

Process Yield %	Device Fault Coverage %										
	90	91	92	93	94	95	96	97	98	99	100
50	6.70	8.04	5.39	4.74	4.07	3.41	2.73	2.08	1.38	0.89	.00
60	4.98	4.48	4.00	3.51	3.02	2.52	2.02	1.52	1.01	0.51	.00
70	3.50	3.16	2.81	2.47	2.12	1.77	1.42	1.08	0.71	0.38	.00
80	2.21	1.99	1.77	1.55	1.33	1.11	0.89	0.67	0.45	0.22	.00
90	1.05	0.94	0.84	0.73	0.63	0.53	0.42	0.32	0.21	0.11	.00

With the increasing complexity of ASIC designs and DFT methodologies, the need for a test architecture based on a structured DFT became essential. The EDA industry responded with a technology, referred to as Test Synthesis, and the tools to provide automatic enhancement of testability based on a behavioral description of the ASIC. The high-level techniques of this technology transform the behavioral description of a design, as written in an HDL, into a structured implementation of data path logic and control logic.

Test synthesis techniques automate the processes of design analysis, insertion of test structures into the design, and post-insertion design analysis. The high fault coverage attainable improves the quality of the ASIC device, and as a result, system reliability (Halliday and Young 1994).

There are failure modes associated with ASICs that are not readily identifiable, since the device functionality cannot be completely simulated. Without engineering and technical test support, defects could easily avoid detection and result in low quality products. A device fault discovered after an operational failure of a safety-critical system could have catastrophic consequences.

Test equipment, synthesis support tools, and procedures used to test ASIC devices must be carefully evaluated. The test procedures and process must be followed and monitored rigorously, and the results documented and verified. The establishment of a good quality management program is absolutely essential to ensure maximum system safety.

A fault coverage of 100 percent is required for safety-critical applications. No one test methodology by itself can guarantee a 100 percent fault coverage, therefore multiple test methodologies must be implemented in the ASIC design (Runyon 1995).

6.2 TEST METHODOLOGIES.

6.2.1 Built-In Self-Test.

Built-in self-test (BIST) essentially allows an ASIC to test itself. Creating a device-level BIST architecture enables the ASIC to perform a complete self-diagnostic test for both manufacturing and field system-level testing. BIST eliminates the need for high performance test equipment.

BIST is accomplished by designing a stimulus generator and a response analyzer into the ASIC. During test, the stimulus generator applies a pattern to the device and the response analyzer gathers the results in the form of a long binary data string, compresses the data into a signature, and compares it to the signature of a good device.

The large quantity of circuits used in an ASIC device makes it difficult to achieve a high percentage of test coverage. The high fault coverage attainable from BIST dramatically improves the quality of the ASIC and therefore the system reliability (Stroud 1991).

Basic logical structures like memories and multiplexers are simple to test with BIST techniques (Strickland 1995). BIST can be configured to produce a fault coverage of greater than 95 percent for memory blocks using only a few gates.

Random combinational logic blocks are difficult to test and may require other techniques to be used in conjunction with BIST in order to achieve the desired fault coverage. Random combinational logic testing is usually achieved by building BIST on top of a scan base test architecture. This test methodology is referred to as ScanBIST.

When a BIST method is being considered for testing non-structured random logic, a thorough and detailed analysis effort is necessary to calculate a good signature. The signature must be stable and repeatable. If the signatures from a good and bad device are identical, or two good devices have different signatures, the test is useless. This condition is referred to as aliasing.

Some of the advantages of BIST testing are as follows:

- Provides high fault coverage.
- Is easy to use and understand (provides conceptual confidence).
- Supports vertical testing (is used at all levels of testing).
- Operates at system speed.
- Enables structured logic testing of memory, etc.

Some of the disadvantages of BIST testing are as follows:

- Requires scan foundation for random logic testing (ScanBIST).
- Requires high silicon overhead (5 to 40 percent).
- Has limited availability of development support tools.

6.2.2 Scan Testing.

Scan-test is the most universally effective test architecture used to increase the manufacturing quality of complex synchronous ASIC devices. The goal of scan-test is to maximize the device fault coverage by designing total controllability and observability into the ASIC device. Implementation of scan test requires that all storage registers, flip-flops, latches, and counters be designed to facilitate scan operation and provide capability for two operating modes, system-mode and test-mode. Figure 2-5 shows how a scan chain can be implemented.

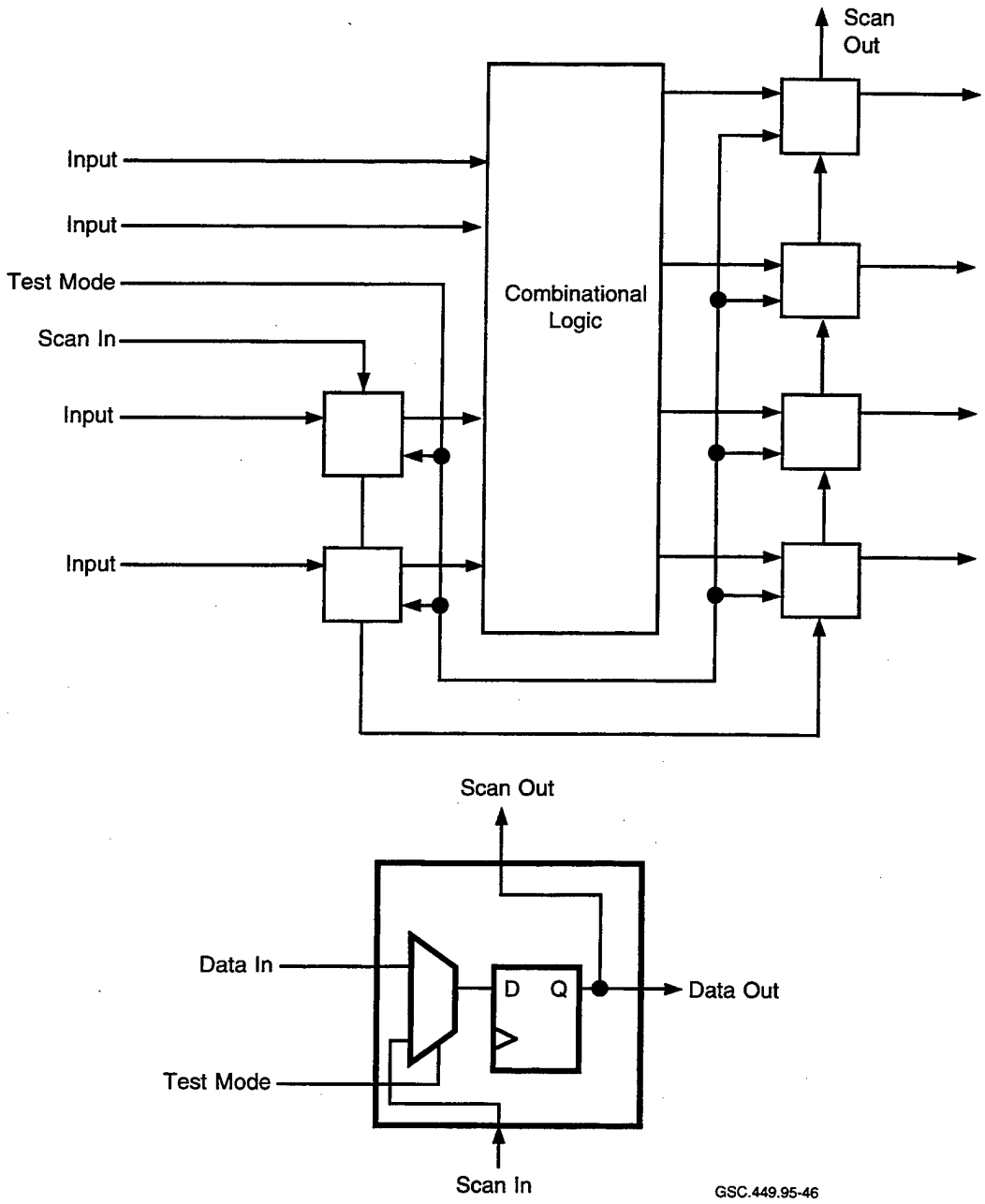


FIGURE 2-5. SCAN CHAIN CONFIGURATION

In the test mode of operation, register inputs and outputs are logically reconfigured and interconnected to form a long shift register, or scan-chain. The shift register scan-chain presents a much simpler circuit form to test than the original complex sequential and combinational circuit configuration. Each register now serves as an input and output to its respective interconnecting internal circuits.

To test the ASIC device, the Test-Mode input is activated and an initializing serial pattern of 1s and 0s are serially clocked into the shift register through the Scan-In terminal. The test vector is clocked through the input combinational logic by the application of a single clock pulse and the results captured by the scan-chain shift register. The data pattern is then compared to the expected good data pattern by the tester. This test sequence of operations is repeated for each test vector to ensure no internal ASIC node is "stuck-at" a logic 1 or 0 state.

Some of the advantages of full scan testing are as follows:

- Highly structured.
- High fault coverage.
- Good fault isolation.
- Easily understood.
- Many EDA tools available.
- Easy to implement and use.

Some of the disadvantages of full scan testing are as follows:

- Large number of test vectors required.
- High test equipment costs (high speed and capacity).
- Works for synchronous logic only.
- Slow; testing done serially.
- High silicon overhead (typically 20 percent).
- No vertical testing capability.
-

6.2.3 Partial Scan Testing.

Partial scan presents a trade-off between the ease of testing and the costs associated with full scan test design. Less substrate area is required for partial scan testing because fewer registers are placed in the scan-chain. Logic synthesis support tools can analyze the testability of the device and predict the number of scan registers needed to achieve a desired fault coverage. During this process, recommendations are made for specific registers to be converted to a form that can be scanned, subject to design and specification constraints.

Scan isolation is a partial scan technique used to test embedded functional blocks of logic, such as a microprocessor core. Scan isolation weaves a scan-chain around all the inputs and outputs of the block for serial access from an external tester or the internal BIST circuitry.

Some of the advantages of partial scan testing are as follows:

- Allows for user-definable test and DFT goals.
- Supports partitioning by best test method.
- Has less silicon overhead than full scan (1 to 15 percent).
- Has less performance impact than full scan.

Some of the disadvantages of partial scan testing are as follows:

- Requires more test vector generation time than with full scan.
- Is difficult to translate vectors for tester use.
- Requires completed design to determine fault coverage and silicon requirements.
- Has limited availability of DFT tools.

6.2.4 Boundary Scan Testing.

Boundary scan test (BST) is based on the IEEE Standard 1149.1 - Test Access Port and Boundary-Scan Architecture, developed for printed circuit board and system-level testing applications.

There are three major components to a BST architecture (Sherman 1995):

- A set of dedicated device terminals for control, clock, and test data, referred to as the test access port (TAP).

- On-chip port controller to direct the test operations, known as the TAP controller.

- A register cell for each device I/O terminal internally connected in a scan chain configuration to form a serial data path, called the boundary scan register (BSR).

BST provides a method for accessing the inputs and outputs of a device directly without making physical tester-to-core logic connections, as shown in figure 2-6. Isolated testability is obtained by the addition of BSR cells placed between each pin of the device and the associated on-chip circuitry. Test data patterns are entered through the test data in terminal and withdrawn serially from the BSR chain at the test data out terminal.

The IEEE 1149.1 specification not only defines the device architecture and protocol, but also defines a Boundary Scan Description Language (BSDL). Through BSDL, the Automatic Test Pattern Generation (ATPG) tools and simulators are told how a device implements BST (Sherman 1995). A critical step in this process is learning how to develop BSDL files and how to validate their accuracy. The BSDL file must describe the design of the device accurately. Even the slightest error can give inaccurate results or damage the device.

The low speed of BST makes it impossible to detect delay, timing, and other speed-related faults. In addition, BST is not suitable for testing dynamic circuits which generate transient or charged output signals that decay with time. BST is not useful for prototype development because it does not have the controllability and observability necessary to debug a design. Also the device must be fully functional, including hardware and software, in order to use BST (Fleming 1990).

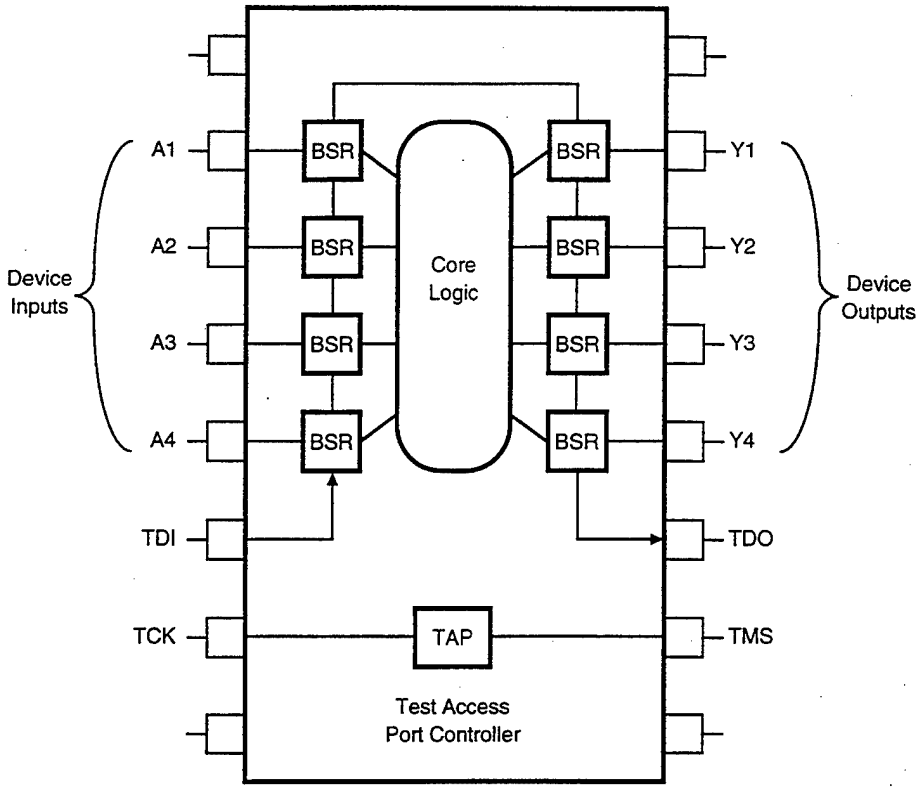


FIGURE 2-6. BOUNDARY SCAN CONFIGURATION

Some of the advantages of BST are as follows:

- Easily understood and implemented.
- Requires low silicon overhead (less than 5 percent).
- Supports high volume production testing.
- Supported by industry standard IEEE 1149.1.
- Supports field and vertical testing capability.

Some of the disadvantages of BST are as follows:

- Low availability of automated test synthesis support tools.
- Not effective for prototype development.
- Not an at-speed test.
- Low fault coverage.

6.2.5 CrossCheck™ Testing.

CrossCheck™ is a non-behavioral, massive embedded observability test methodology developed and patented by CrossCheck Technology Inc. of San Jose, California. The entire CrossCheck test structure, as shown in figure 2-7, is completely transparent to the ASIC designer (Levitt 1992).

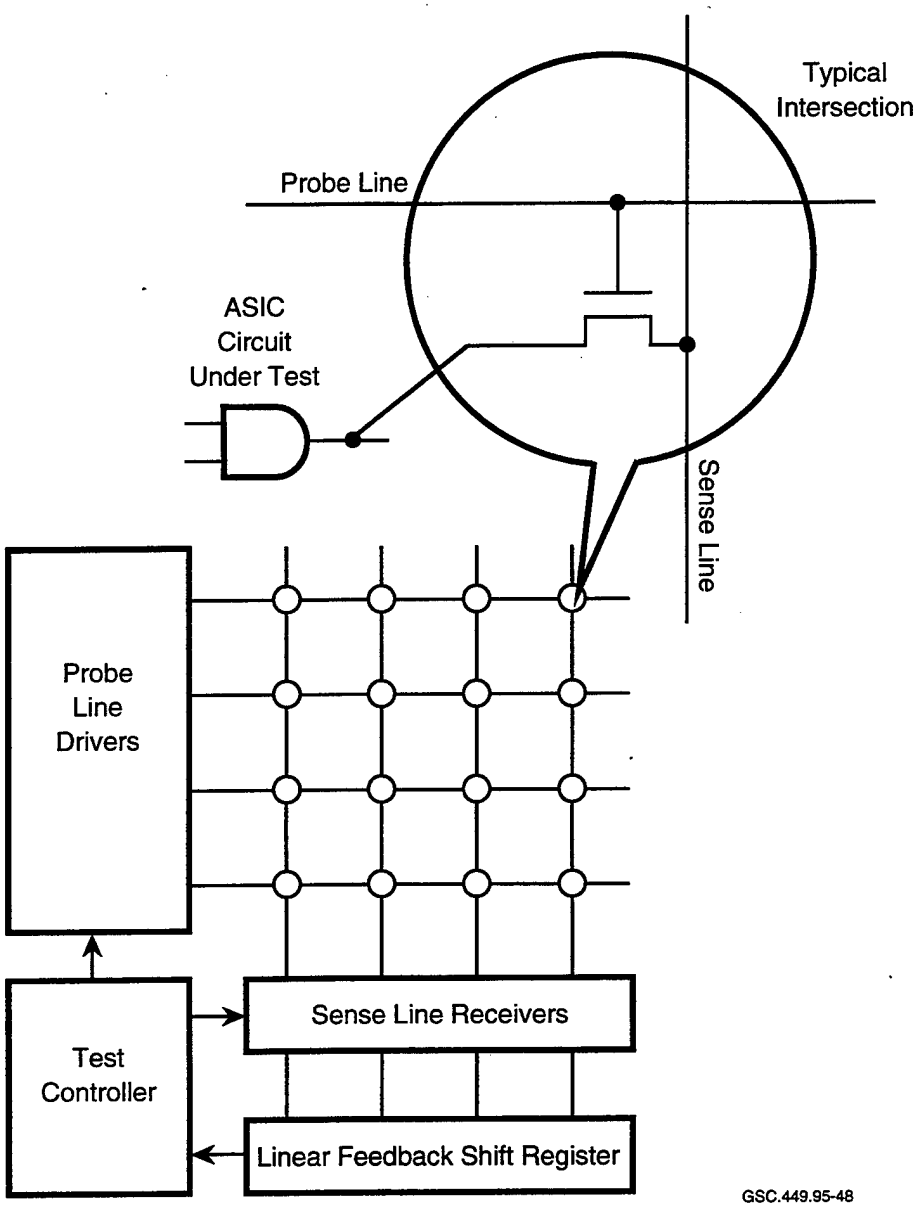


FIGURE 2-7. CROSSCHECK CONFIGURATION
(Levitt 1992)

An overlaying grid of built-in test points are added to an ASIC's base array. Each test point is a CMOS transistor that allows the output value of a gate to be transferred to a sense line when a corresponding probe line is selected by an associated probe line driver (Lorusso and Fertsch 1991). The sense lines connect to an analog sense-amplifier which conducts a voltage level analysis to determine if fabrication defects are present. The output of the sense amplifier is connected to a Linear Feedback Shift Register (LFSR) which compresses the test data into a signature. The signature is shifted out to the tester for verification.

Some of the advantages of CrossCheck testing are as follows:

- Provides extremely high fault coverage.
- Requires no design-in effort.
- Supported by development and simulation tools.
- Produces little or no performance impact.
- Supports vertical testing through BST.

Some of the disadvantages of CrossCheck testing are as follows:

- Slow (not at-speed).
- Requires moderate silicon support (typically 15 percent).
- Limited support (patented process).

6.2.6 I_{DDQ} Testing.

I_{DDQ} testing methodology is a massive observability technique that will not only detect failures, but will also identify leakage currents that could cause the ASIC device to fail in service. I_{DDQ} testing detects ASIC faults by measuring the device quiescent power supply current.

Accurate off-chip I_{DD} measurements are not always easy to acquire using existing test equipment, because the resolution of these low current measurements is limited. Accurate on-chip I_{DD} measurements can be acquired using on-chip built-in current sensors (BICS) to detect abnormal power bus currents. On-chip current measurement resolution can detect much smaller values of abnormal current and allow I_{DDQ} testing to be operated at a much higher rate of speed than is possible with off-chip measurement techniques (McEuen 1992).

I_{DDQ} testing has the greatest impact on quality and cost when performed at the substrate level. I_{DDQ} tests should be repeated after the burn-in cycle which activates hidden defects. These tests are most effective with V_{DD} operating voltage set to the maximum specified value.

For high reliability devices, as required for safety-critical applications, the test program should include I_{DDQ} testing. Experimental data shows that defect detection increased between 60 and 80 percent when I_{DDQ} testing was implemented in the test program (Hawkins et al. 1992). Automotive industry studies show that about 10 percent of the faults are detectable only using I_{DDQ} testing (Runyon 1995).

Some of the advantages of I_{DDQ} testing are as follows:

- Detects reliability problems.
- Provides high fault coverage.
- Requires little or no silicon support.
- Reduces the number of circuits and test vectors required.

Some of the disadvantages of I_{DDQ} testing are as follows:

- Requires isolation of circuits that draw static current.
- Has low to moderate speed operation (100 kHz to 1 MHz).
- Has poor availability of development support tools.

6.2.7 Ad Hoc Testing.

Ad hoc testing is a nonbehavioral testing technique. For this testing methodology to succeed, each element of the ASIC must exhibit controllability and observability. These properties are achieved by imposing design rules tailored to each circuit to be tested. Ad hoc testing techniques are implemented successfully when gate counts are under 10,000. Considering the current complexity of ASIC devices, this technique no longer appears to be a viable testing approach.

Some of the advantages of ad hoc testing are as follows:

- Good random logic test capability.
- Low silicon overhead (less than 5 percent).
- Minimal device performance impact.

Some of the disadvantages of ad hoc testing are as follows:

- Requires manual test pattern generation.
- Increases device I/O terminals.
- Has no availability of CAD or development support tools.

6.2.8 Behavioral Testing.

Behavioral testing tests the ASIC device in the same way as the device is operated in the system. All specified functional operations and associated test data vectors are generated by the tester. In the system, the ASIC could be exposed to many conditions and unspecified sequences of operation. To test for all these combinations would be impractical if not impossible, therefore behavioral testing by itself cannot achieve 100 percent fault coverage (Strickland 1995).

Also, there are failure modes that can only be detected with behavioral at-device-speed testing methods. Since behavioral tests are operated at-system-speed, which generally is slower than at-device-speed, these speed-related faults will not be detected in test.

7. APPLICATION SPECIFIC INTEGRATED CIRCUIT VERIFICATION AND VALIDATION.

7.1 DIFFICULTIES IN VERIFICATION.

According to Zaidi (1995) "the most difficult task in system-on-a-chip design is verification of system functionality." There are no standard methods for verification and validation (V&V) of ASICs. Designs using different technologies, architectures, and design techniques require that

V&V be tailored to each specific application. A problem with V&V is that it can be haphazard and random, if not planned, executed, and monitored thoroughly from the start of the design cycle.

Rapidly evolving technology requires that verification methods also change. Simulation is one of the main verification methods for ASICs. A major problem that verification engineers face is that simulation speed can be very slow. Special tools are sometimes designed in order to speed up specific simulation tasks.

With any complex verification task, developers hope that all the design errors can be found, but this is not always the case, nor does it mean that passing verification testing results in an error-free design. Verification testing needs to account for the ASIC architecture. Random test vectors are applied, but are limited in their ability to detect problems. Controllability and observability of internal logic becomes less visible as more functions are integrated into ASICs. Special verification tests suited to the unique characteristics of each ASIC design are necessary.

Each step in the design, test, and fabrication cycle is equally important. A weak link in any one area is an invitation for disaster. This was demonstrated by the recently discovered Pentium IC flaw. For the Pentium IC, a different technique was used for floating point division than was used on the 80486 CPU. The technique, called "radix 4 SRT," computed the quotient twice as fast as the conventional shift-and-subtract method, since it generated two bits of quotient for every clock cycle. A plot of partial remainders versus divisors is used to access the next quotient digit by means of a lookup table.

The lookup table was generated manually during the development process, externally to the Pentium IC. In order for the table to be loaded into the hardware, a script is used. A problem in the script caused the inadvertent omission of several table entries. Consequently, any access of the lookup table at these table offsets generated the floating point division error.

While ASICs with design errors that make it into commercial products are not uncommon, this one in particular was the "most widely publicized bug in computer history" (Geppert 1995). Considering the financial consequences for Intel (nearly one-half of a billion dollars), this bug will be remembered for a long time. Are bugs such as this simply a way of life for designs that are complex and difficult to verify? According to Levy (1995):

> ...when you're designing a device with millions of transistors, the chance of reaching absolute perfection is minimal... The question is: When and how will a bug surface?

The Pentium problem demonstrates that each part of the design and development process is important to the integrity of the final product. Not only is it imperative to verify that a design is correct, but also that tools are used correctly, and that there are no interface problems between tools, or database problems. Editors, compilers, HDLs, synthesizers, placement and routing tools, file conversion tools, and download tools need to be able to pass database information accurately and need to have compatible data formats.

7.2 SIMULATION.

There are a number of simulation approaches used for verification. Traditional simulation is event-driven and uses interpretation of the HDL, and not compilation, for model execution. Every signal value change is scheduled by the simulator and simulation may be paused between any two events. This approach allows a great deal of flexibility for design debugging. However, extra overhead associated with event handling is incurred as a result of this flexibility. Complexity has increased functionality and the number of events which must be processed, resulting in increasingly higher run-times required for simulation.

Essentially, a simulation executes a model of a real device. Models have been generated and simulations run on device designs from the specification level down to the gate level. Simulation is relied upon to flag errors and verify correct operation at all design levels. Simulation allows rapid design iterations, so that changes can be tested easily. Due to the cost involved in rework, detection of errors before the silicon is cast is a high priority for any development effort. Accurate simulations are relied upon to meet these demands.

Although simulation allows rapid design iterations, it is fundamentally a verification technique, not a design technique. While simulators are an integral part of the design environment, they cannot assist designers in the transition from concept to schematic, generate timing or interface specifications, or demonstrate how to fix a timing problem.

7.2.1 Simulation and Verification.

The Digital Systems Validation Handbook-Volume II, DOT/FAA/CT-88/10 defines verification thus:

> The act of reviewing, inspecting, testing, checking, auditing, or otherwise establishing and documenting whether or not items, processes, services, or documents conform to specified requirements.

For ASIC design, one way this is done is to examine two products of the design cycle and determine whether or not they are identical in some set of specified properties. Often, this comparison is done on a cycle-by-cycle basis. After a design is synthesized, the comparison can be made between the gate-level implementation and the HDL description. These two descriptions can be compared for functional equivalence.

Tools are important for the verification process. Since there is a massive amount of data with which the designers must work, tools are relied upon to automate this task. Tools should automate the functional equivalence comparison. They should allow designers to import and use standard timing constraint information. They should contain tolerance windows which facilitate error identification. Also, the test vectors used for the design verification process should be exportable so that they can also be used for verification of the silicon.

7.2.2 Simulation and Validation.

A general definition for validation is given in the glossary of the Digital Systems Validation Handbook-Volume II, DOT/FAA/CT-88/10. It is defined as

> The process of evaluating whether or not items, processes, services, or documents accomplish their intended purpose in their operating environment.

For ICs, validation involves establishing proof that the design is what the design team intended to capture. It demonstrates that there are no missing, incorrect, or undesired functions. Validation can be done at a number of different levels. At the HDL level, a design function can be validated by simulation. A gate-level design is necessary to validate speed and area design criteria.

For HDL modules, typically only the functional validation is performed. Simulation is used almost exclusively for validation of today's designs. The input for this simulation can originate at a number of different sources, including VHDL and text files. Later, when verification of the post-synthesis functional design is performed, validation patterns are again applied. This can be a problem, since different tools may have different input formats and requirements. Inexact mapping can also be a source of error introduction.

High-level simulation typically consists of iterations of design and debug cycles. Tools need to be concise in reporting errors. Problems should be identified as clearly as possible. When designing with HDL, a source-level debugger should be used, as is done for many high-level languages. Also, a view of the gate-level design is helpful when correcting errors. This aids the designer in understanding what hardware is produced as a result of HDL coding and changes to that coding.

As with other high-level development environments, the simulation environment should be integrated with other related tools, such as the gate-level design and the source-level debugger. Tools should allow rapid movement between these environments so that the effects of changes can be determined rapidly. Error messages, environmental data such as the directory tree, simulation results, and other such messages should be displayed appropriately.

When the synthesis is performed, validation is necessary at the gate level. A large number of input vectors is needed to obtain reasonable levels of coverage. One of the important considerations is simulation speed. Faster simulation with test vectors allows more of the design to be validated. Methods used to speed simulation include hardware acceleration and the use of multiple CPUs.

7.3 APPLICATION SPECIFIC INTEGRATED CIRCUIT PROTOTYPING AND VALIDATION.

Designers should seriously consider prototyping when there is a long build time for the silicon, or there are dire financial or safety issues. Whether to prototype or not should not be based on the probability of an error in the logic, but rather its consequences if undetected. In assessing risk, following are pertinent questions:

- Will correct operation be ensured if the design is implemented without logical errors?
- Are there submodules in the design that have not been used previously?

- As data move through the submodules, do they follow easily testable bounds?
- Are data understood and testable between and at the boundary conditions?
- Which will cost less, a redesign or a prototype?
- Does the schedule allow time for a redesign?

Since simulation does not verify correct design and operation adequately in certain cases, prototyping can be an effective addition to the verification suite. For safety-critical systems, prototyping should be a requirement.

7.4 FORMAL METHODS FOR HARDWARE DESIGN.

Formal verification uses an analytical approach for proof of correctness and does not rely on the use of test vectors, as with simulation. The method is inherently thorough, as opposed to test vectors, which in practice do not provide full coverage. Many tools now incorporate formal methods so that this verification technique does not need to be performed as a separate, manually intensive task.

Formal verification takes only minutes to execute on current tools and checking small revisions is not a problem for this technique, as it is for simulation. Formal verification results highlight faulty logic, whereas simulation does not. Formal verification also facilitates verification of modules. A drawback of formal verification is that it does not verify timing. It is best to use formal verification along with a static timing-analysis tool.

7.5 SOFTWARE REUSE.

Software reuse is an issue for designers. Once code is developed, it may be possible to use large portions of the code in a different application. This is easily done for software, since code is portable. Hardware, on the other hand, has not shared this capability. However, with the use of HDLs, logic designs, or portions thereof, can be reused. In fact, when ASIC designs contain tens of thousands of lines of code, there is significant motivation for developers to reuse as large a portion of that code as possible. Development cost can be reduced and the TTM shortened for products where HDL reuse is implemented.

For software, certification credit has been a consideration for code that had been certificated previously as part of an avionic system. A question arises as to whether digital hardware designs will be treated in the same manner as software. For instance, since current complex ASIC designs are essentially software projects, should portions of the HDL code that have been certificated previously as part of a larger system, receive credit for that certification?

7.6 REGULATIONS AND GUIDELINES.

On the regulatory and design guidance side, the Federal Aviation Regulations (FARs), RTCA/DO-160C - Environmental Conditions and Test Procedures for Airborne Equipment, SAE ARP1834 - Fault/Failure Analysis for Digital Systems, and Advisory Circulars (ACs), address issues from specific perspectives.

RTCA/DO-160C addresses issues relating to the environment in which airborne equipment must operate. While the RTCA/DO-160C testing guideline is necessary, it is not the only guideline that can be applied to Line Replaceable Units (LRUs). Failure analysis on avionic equipment can be accomplished using procedures in ARP1834. The need for failure analysis techniques is pointed out in AC 25.1309-1A and FAR Parts 23, 25, 27, and 29, section 1309, and is implied by FAR Part 33, section 75. ARP1834 has been adopted as an informal guideline for meeting these requirements.

ARP1834's analyses are specifically meant to identify digital equipment hardware faults. ARP1834 is not an exhaustive or universally accepted method for applying fault/failure analysis. It is used merely to present cost-effective, industry-acceptable means for identifying failure modes and failure effects. Manufacturers who wish to use ARP1834 as a certification guideline should discuss their reasoning with the regulatory agency early in the process. This is because variations of approaches presented in ARP1834 will need to be employed under different circumstances. For systems that are flight-critical or flight-essential in nature, one approach might be to develop design techniques for a fault tolerant system.

However, testing to RTCA/DO-160C and ARP1834 is not, by itself, sufficient to ensure that failures will be extremely improbable, as required for flight-critical systems. While avionics manufacturers and airframers do use additional tests and design assurance methods, currently there are no guidelines that address the life-cycle considerations for complex ICs. Thus, the certification process lacks uniformity. What is deemed sufficient by one Aircraft Certification Office (ACO) may not meet the demands of another ACO. Additionally, manufacturers need to know what to expect during the certification process.

While it is obviously in the best interest of the avionics supplier to provide reliable equipment, the methods manufacturers use vary. Reliability and safety issues, as applied to the development processes of complex ICs designed for safety-critical applications, are in dire need of being addressed. In an effort to standardize the system certification process for hardware, and address current hardware issues, RTCA has formed Special Committee 180 (SC-180).

8. CONCLUSION.

Current fly-by-wire technology necessitates greater concern for safety issues relating to digital flight control and avionic systems. Fly-by-wire aircraft are using ASICs to implement flight-essential and critical functions. Threats to device reliability exist and need to be addressed in complex IC-designs. These complex ICs include devices such as FPGAs and ASICs. While classically the ASIC was not user-programmable, in essence, it now is. An engineer with the right tools on a PC can do the entire design, simulation, and test generation, and then send it to the silicon foundry for fabrication and packaging.

There are many advantages to be gained by using complex user-programmable devices, such as ASICs, but a closer examination is warranted when flight-critical or -essential systems with user-programmed complex ICs are presented for certification. It is not a simple matter to demonstrate that these devices are designed correctly or tested adequately.

8.1 A NEW LOOK AT AN OLD TECHNOLOGY.

The use of digital technology in aircraft is nothing new. Implementations of early digital logic ICs were relatively easy to analyze and their failure modes were well understood. While the use of ASICs in aircraft is seen as a benefit for avionics manufacturers and airframers, its implementation raises concerns about the safety of systems in which they are used. Part of the problem is due to the sheer complexity of current ASIC devices. Failure analysis guidelines that were developed for digital systems, such as SAE's ARP1834, cannot be applied in a meaningful way to the complex ICs that are being designed today. Additionally, new failure modes, that were not a problem for older digital technology are now prevalent and can compromise the safety of systems using these complex devices.

Commercial fly-by-wire aircraft are now being produced that have differing design philosophies from earlier aircraft. While digital avionics and flight controls have existed for a number of years, there were always backup systems that relied on a different technology, in case of a failure of the digital system. These were hybrid aircraft, using a combination of control hydraulics with interfaces to digital systems. Today's fly-by-wire aircraft, as typified by the Boeing 777 and Airbus A320, use data buses to send actuation commands to the various control surfaces, based on messages generated from the avionic systems. On these aircraft, the hydraulic link no longer exists. It is expected that if there is a failure on a primary system, another system with duplicate capabilities and connectivity will be available to take control. Back-up systems are simply duplicates of the primary systems. Redundancy may sometimes be implemented using dissimilar hardware and software, and integrity enhanced by voting systems and other techniques, but the technology remains the same.

Due to increased IC densities, ASICs can now be programmed to take on tasks that were formerly performed in software. For instance, communication protocols are implemented in an IC. High-Level Data Link Control (HDLC), and ARINC 629 are complex transactions that are described in a written specification, and implemented in silicon. It is no trivial matter to verify (1) that the protocol is completely and correctly specified in written form, (2) that it was implemented completely and correctly in silicon, and (3) that the silicon is not defective. At current ASIC complexity levels, it is dangerous to assume that upset avionics are due solely to software bugs.

Error-free ASICs can no more be guaranteed than one can promise error-free software. In fact, complex ASIC design is described by Corcoran (1995) as a "software project being performed by hardware engineers," since most ASIC parts are now designed using high-level languages that describe digital logic behavior.

The following sections will highlight and discuss some of the problem areas that may be encountered when working with complex ASICs.

8.2 CERTIFICATION.

It is anticipated that almost all hardware in the future will be composed of ASICs, FPGAs, and other user-programmable and special function ICs. Sound engineering practice necessitates the

development and use of a process to guide design, development, and test toward meeting specific design requirements and safety goals.

ASICs have been used by airframers to circumvent the rigors of the software approval process (Shaw, Herzog, and Okubo 1986). Avoiding scrutiny for a flight-critical system component by seeking a hardware solution that is less encumbered by regulations is a way for developers to cut costs, but the overall safety impact should be examined carefully. Fundamental verification issues can be bypassed with a silicon-based implementation.

Certification of systems containing ASICs is a potential problem for developers and CEs. Currently, there are no techniques and methods of design, documentation, testing, and verification identified or recognized by the FAA for today's complex hardware designs. Existing guidance does not address current practice or technology. For instance, ASIC designs are implemented using a full suite of computer-based tools that are not regulated by any guidelines, while tools used in software development are. (RTCA/DO-178B requires tool qualification for software tools in safety-critical applications.)

Work is underway through RTCA's SC-180 to develop hardware design assurance guidance for digital systems, although it is not anticipated that the resulting guidance will address all the issues. Questions concerning hardware/software integration, hardware reuse, hardware/software codevelopment, and other issues are likely to remain. While process guidance is necessary, certification specialists should keep in mind that new testing and design verification methods are still emerging in this rapidly evolving technology. New methods that enhance device controllability and observability, along with more rigorous verification methods should be encouraged.

8.3 DESIGN.

With ASIC technology, design and coding is not an isolated function, but relates to each step in the development and test phases. A change necessitated by simulation, synthesis, test, or layout necessitates a change in the HDL code. Complex ASIC design is not really hardware design as it once existed. ASICs are now designed using software. VHDL is a structured language, and coding proficiency is not achieved easily. A complex ASIC can require a hundred thousand lines of high-level code. In the future, as complexity increases, this number will grow significantly.

As a rule-of-thumb, designs with more than 10,000 gates are not done by schematic entry. Tools become a necessity above this level. Due to constant rapid developments in digital technology, however, tools are still playing catch-up to the technology, leaving designers with little support in some areas of design and test. Tool-induced design errors occur and can be difficult to detect. The fidelity and completeness of simulation tools is essential to design integrity, yet modeling deep-submicron phenomena is an area that lags behind current technology. Being able to predict accurately signal propagation delays can mean the difference between a properly operating device, and one that is marginal or failing.

8.4 TESTING AND VERIFICATION.

Thorough device testing is critical to the quality of an ASIC and, for safety-critical systems, to the safety of the user. The goal for test is to ensure that no faults exist which can cause the device to malfunction. Any limitation in test capability can mask problems and allow defective devices to be installed in a system which eventually may fail in operation. If the failure is in a safety-critical system, the effects must be noted and corrected by the system or system operator, or the results could be disastrous to both life and property.

Automotive and medical electronics companies demand 100 percent fault coverage for ASICs used in vehicle control and human life support systems. Demanding a fault coverage of 100 percent for ASICs used in safety-critical avionics is only the beginning in achieving safety. A good quality management program that will follow the ASIC development from design through installation and maintenance is essential. This program includes personnel, development and test equipment, supporting systems, procedures, documentation, environment, and the ASIC devices.

Quality and safety begin in design. To achieve 100 percent fault coverage, all ASIC faults must be detectable in manufacturing test. Many circuits can be tested by performing at-system-speed test diagnostics that simulate the way the device is used in the system. This behavioral testing method generally can provide 60 to 80 percent fault coverage. The remaining circuits are either impractical or impossible to test using this method and are tested using non-behavioral DFT methods which are designed into the ASIC.

Several well-supported DFT methodologies are currently available. Each has its own set of advantages and disadvantages to consider. Which is best to use depends on the requirements and design specifications of the particular ASIC being developed. If no single testing method will provide 100 percent fault coverage, multiple techniques must be used. One method that always should be used in testing complex ASICs is quiescent current (I_{DDQ}) testing. Approximately 10 percent of all ASIC faults are detectable only with I_{DDQ} testing.

Test synthesis tools are providing the capability for anyone knowing the application to design an ASIC using functional descriptive techniques. Meaningful verification may be difficult to demonstrate since knowledge of all development steps and their associated products is necessary. In order to detect and identify ASIC faults, IC engineers and process specialists must be part of the development team. Certain faults, resulting from the existence of physical phenomena in the ASIC, are undetectable and unknown to the designers during synthesis. Generally, these faults can be detected using at-device-speed behavioral test methods.

Even ICs, whose failure can cause substantial financial penalties to the manufacturer, such as the Intel Pentium, are not immune from process errors that can cause defective hardware. There was no design error in Intel's Pentium microprocessor, but a step in the development process had allowed bad data to slip into the design. Even seemingly insignificant process steps that are well understood and rarely have been the source of problems can be a source of failure. This error cost Intel close to $500,000,000.00 (Geppert 1995). It serves to show that errors sometimes do find their way into

hardware, even though this hardware underwent some of the most rigorous verification processes in industry.

Unlike software, ICs are influenced by changes in temperature, voltage, noise, and other environmental variables. Combinations of fluctuations in these variables can induce failure. Temperature changes can produce timing skew; voltage changes produce temperature changes and also change noise immunity characteristics. Thorough device testing within the manufacturer's specified operating environment is impractical.

There are failure modes associated with ASICs that are not readily identifiable. They can be the result of design errors or subtle phenomena that are not flagged by the tool suite. These phenomena include clock skew, ground bounce, and crosstalk. ASICs can exhibit data-sensitive behavior due to the cumulative effects of internal currents resulting from peculiar data patterns. These errors may not be found during testing due to the impracticality of complete pattern testing for a complex ASIC. If these ASICs are part of an avionic system, they may increase the number of LRUs removed for servicing and lead to more "no fault found" results at the test bench.

It can be difficult to detect faulty operation of an ASIC. Complex ASICs require that test circuitry be designed into the ASIC. Designs for fly-by-wire aircraft require fault tolerant architectures, not simply redundancy. There are no FAA guidelines that would suggest to airframers how this is done, or what to require of their avionics suppliers.

Due to the current necessity to create ASICs and FPGAs using software (i.e., at a much higher level of design abstraction than in the past), systems engineers and even programmers are designing ASICs. Meaningful verification may be difficult to demonstrate since knowledge of all development steps and their associated products is necessary.

8.5 COMPLEXITY ISSUES.

According to Keller (1992)

> Some electronics systems themselves have attained a complexity level such that traditional lab and flight testing methods cannot realistically be extensive enough to show compliance to existing requirements.

This is certainly true for ASICs, which are at the core of modern avionics. ASICs can contain embedded microprocessor cores with user-supplied software. Complex ASICs can replace complete systems.

Typically ASICs are programmed by the end-user or avionics supplier. Complex ASIC designs can require teams of 50 to 100 engineers. ASICs are a technology essentially unregulated by the FAA, and not understood by CEs. The level of on-chip circuit elements that can be squeezed into an ASIC is so high that more of the software portions of avionic system designs are being placed into ASICs. ASICs are used extensively on the Boeing 777 in flight-critical systems and, along with

other user-programmable logic and special function ICs, will be used almost exclusively in future fly-by-wire aircraft.

Predicting reliability for ASICs involves more than simply predicting the probability of a hardware failure. Classic reliability figures assume that the device has no design errors and that the silicon has no defects. For complex ASICs, these assumptions may not be true. ASIC designers cannot guarantee error-free designs, and ASIC manufacturers cannot guarantee error-free silicon. ASICs used in flight-critical systems can contain latent faults, having unpredictable effects.

Another issue which complicates reliability prediction is the fact that complex ASICs are now created using high-level software. In order to arrive at a more accurate estimate of ASIC reliability, software reliability issues may also need to be taken into account.

Some may feel that in order to avoid the burgeoning complexity and potential problems associated with deep-submicron design, this technology should be avoided altogether for safety-critical systems and only larger ASIC geometries should be used. This approach may be possible, but only for a little while. As the semiconductor industry continues to invest in new technologies, there comes a point when the older technologies are no longer supported simply due to economic considerations. This obsolescence seems to be occurring at a rapid pace. It is therefore doubtful that avoiding deep-submicron design could be anything more than a short-term avoidance of an inevitable problem.

ICs eventually fail. Therefore system architectures will always include redundancy. Fault tolerant architectures should be in place, not only at the systems level, but also internal to the ASICs. In order to manage design complexity and ensure that each internal functional block can be tested thoroughly, sufficient partitioning of internal logic needs to be included. Architectures that facilitate fault identification such as parity, watchdog timers, variable limit checking, and other such schemes should be in place to flag incorrect device behavior.

9. BIBLIOGRAPHY.

ARP1834, Fault/Failure Analysis for Digital Systems and Equipment, Society of Automotive Engineers, Warrendale, PA, August 7, 1986.

Baze, M.P., et al., "Single Event Upset Test Structures for Digital CMOS Application Specific Integrated Circuits," IEEE Transactions on Nuclear Science, Volume 40, No. 6, December 1993.

Beechler, Robert K. and Robert Hanz, "The Role of Vendor-Specific Programmable Logic Tools," ECN, Volume 38, No. 10, October 1994.

Corcoran, Tim, "Top Down ASIC HDL Modeling Methodology," On-Chip System Design Conference, Day 2, Hewlett Packard, 1995.

Coston, Terry W., "A Status Report on Analog HDLs," Integrated System Design, Los Altos, CA, November 1994.

Digital Systems Validation Handbook-Volume II, DOT/FAA/CT-88/10, U.S. Department of Transportation, Federal Aviation Administration, February 1989.

Donlin, Mike, "Standards Groups Vow to Solve Tool Interoperability Problems," Computer Design, Volume 34, No. 6, June 1995.

DOT/FAA/AR-95/31, Design, Test, and Certification Issues for Complex Integrated Circuits, U.S. Department of Transportation, Federal Aviation Administration, January 1996.

Fleming, Peter, "Applications of IEEE Std. 1149.1: An Overview," The Test Access Port and Boundary Scan Architecture, IEEE Computer Society Press, Los Alamitos, CA, 1990.

Geppert, Linda, "Biology 101 on the Internet: Dissecting the Pentium Bug," IEEE Spectrum, February 1995.

Gruebel, Robert, "Using DFT in ASICS," EE-Evaluation Engineering, Volume 34, No. 3, March 1995.

Halliday, Andy and Greg Young, "Test Synthesis - System to Foundry," Test Synthesis Seminar - Digest of Papers, IEEE International Test Conference, Altoona, PA, October 1994.

Hawkins, Charles E., "I_{DDQ} Design and Test Advantages Propel Industry," IEEE Design & Test of Computers, Volume 12, No. 2, Summer 1995.

Hawkins, Charles E., et al., "Quiescent Power Supply Current Measurement for CMOS IC Defect Detection," Bridging Faults and I_{DDQ} Testing, IEEE Computer Society Press, Los Alamitos, CA, December 1992.

IEEE Standard 1149.1, Test Access Port and Boundary-Scan Architecture, Institute of Electrical and Electronic Engineers, May 1990.

Janowitz, Joan, "Handbook-Volume III Digital Systems Validation Book Plan," DOT/FAA/CT-93/16, 1993.

Kapusta, Richard, "Options Dot the Programmable Logic Landscape," EDN, Volume 40, No. 14, July 1995.

Keller, Forrest L., "Basic Electronic Systems Certification," Proceedings of the IEEE/AIAA 11th Digital Avionics Systems Conference, October 1992.

Keller, John, "Airframers Grapple with SEU Problem in Avionics," Military and Aerospace Electronics, July 1993.

Levitt, Marc E., "ASIC Testing Upgraded," IEEE Spectrum, May 1992.

Levy, Markus, "Intel Cures the Pentium Blues," EDN, Volume 40, No. 2, January 1995.

Lorusso, Stephen M., and Michael T. Fertsch, "Integrating CrossCheck Technology Into the Raytheon Test Environment," Test: Faster, Better, Sooner, IEEE International Test Conference, Altoona, PA, 1991.

McEuen, Stephen D, "I_{DDQ} Benefits," Bridging Faults and I_{DDQ} Testing, IEEE Computer Society Press, Los Alamitos, CA, December 1992.

Munson, John C. and Ruth H. Ravenel, "Designing Reliable Software," Fourth International Symposium on Software Reliability Engineering, IEEE Computer Society Press, Los Alamitos, CA, 1993.

RTCA/DO-160C, "Environmental Conditions and Test Procedures for Airborne Equipment," Radio Technical Commission for Aeronautics, Washington, DC, December 1989.

RTCA/DO-178B, "Software Considerations in Airborne Systems and Equipment Certification," Radio Technical Commission for Aeronautics, Washington, DC, December 1992.

Runyon, Stan, "I_{DDQ}: Does it work?" Electronic Engineering Times, August 1995.

Shaw, John L., Hans K. Herzog, and Kenji Okubo, "Digital Autonomous Terminal Access Communication (DATAC)," Proceedings of the IEEE/AIAA 7th Digital Avionics Systems Conference, October 1986.

Sherman, Jeffery, "To Use BSDL Successfully, Validate Device Descriptions Carefully," EDN, Volume 40, No. 12, June 1995.

Strickland, T., "Design-for-Test Methodologies and Tools for ASIC Design," Electronic Products, Volume 38, No. 1, June 1995.

Stroud, Charles E., "Built-In Self-Test for High-Speed Data-Path Circuitry," Test: Faster, Better, Sooner, IEEE International Test Conference, Altoona, PA, 1991.

Waller, Larry, "A Tale of Two ASICs," Integrated System Design, Los Altos, CA, February 1995.

Widman Associates "Introduction to Designing for Synthesis," Seminar, Wescon 94 Idea/Microelectronics Conference, Anaheim Convention Center, Anaheim, CA, September 1994.

Zaidi, S. Jauher A., "System-On-Chip Design and Verification in Mass Storage," On-Chip System Design Conference, Day 1, Hewlett Packard, 1995.

GLOSSARY

ALIASING. A condition where good signatures generated by the test data compression process are indistinguishable from the bad signatures. Aliasing is generally caused by an error in the data compression algorithm.

APPLICATION SPECIFIC INTEGRATED CIRCUIT. A semi-custom chip used in a specific application. ASICs are designed by integrating standard cells and arrays from a library.

ASYNCHRONOUS. 1. Describing a sequential logic system wherein operations are not synchronized to a common clock. 2. Signals whose behavior and timing are unrelated to a particular clock. Signals are based on known but random events whose timing cannot be precisely predicted.

AUTOMATIC TEST PATTERN GENERATOR. A software program that, given a circuit description and a list of faults, automatically generates the test vectors necessary to detect the faults specified for that circuit.

BEHAVIORAL TESTING. A method of testing that exercises the device in the same way it will be used in the target system.

BOUNDARY REGISTER. A register added to cells to combine signals at the boundary between core modules or core modules and the I/O terminals. The boundary register collects data from and presents data to modules on the boundary during test.

BUILT-IN SELF-TEST. A design method that allows a device to test itself by adding logic for test signal generation and analysis of test results.

CHIP. A single piece of semiconductor material which contains integrated circuitry. A chip is also referred to as a substrate or die.

CLOCK SKEW. A variation in the arrival time of the active clock edge between two or more clocked flip-flops. Clock skew can result in incorrect logic values from the affected circuits.

COMPLEMENTARY METAL OXIDE SEMICONDUCTOR. An integrated circuit used for memory and logic cells. It uses negative-well MOS (NMOS) and positive-well MOS (PMOS) transistors configured in a complementary pair fashion that results in low power operation.

COMPUTER-AIDED DESIGN. Specialized software used for architectural, mechanical, or electrical design.

CONTROLLABILITY. The ability to set a node inside a device to a desired value via an input pin. For a digital circuit, the value would correspond to a logic 1 or 0.

CROSSTALK. The unwanted capacitive or inductive coupling of signals between adjacent conductors or circuits.

DATA PATH. A portion of a design that typically comprises arithmetic and word-wide logical operations.

DESIGN FOR TESTABILITY. The design of a device to enhance its controllability and observability and thereby ease test generation.

DIE. Same as a chip, particularly before being placed in an IC package.

ELECTRONIC DESIGN AUTOMATION. Software and hardware tools used to ascertain the viability of an electronic design. These tools perform simulation, synthesis, verification analysis, and testing of the design.

ELECTROSTATIC DISCHARGE. The natural physical event of the transferring of electrical charges. If uncontrolled, ESD can destroy semiconductor devices which have inadequate packaging and handling protection.

FAULT COVERAGE. The number of detectable device faults divided by the total number of possible device faults expressed as a percentage.

FAULT DETECTION. The ability to determine if a fault is present.

FIELD PROGRAMMABLE GATE ARRAY. A logic device that is programmable and has a high density of gates.

FLIP-FLOP. 1. A bistable digital circuit. 2. An electronic circuit having two stable states and two inputs corresponding to the two states. The circuit remains in one state until caused to change to the other by the application of the corresponding input signal. The two states are referred to as the set and reset state, or a logic 1 and logic 0 state.

FORMAL VERIFICATION. Verifying electronic circuit functionality using mathematical proofs.

GROUND BOUNCE. A ringing (damped oscillation) on an output signal when one or more outputs on the same device switch from logic 1 to logic 0.

HARDWARE DESCRIPTION LANGUAGE. A specialized programming language that describes the physical design, electronic behavior, logic structure, and system annotation information for circuits. The language allows design description at a high level of abstraction while supporting a logical synthesis path to gate level implementation.

INTEGRATED CIRCUIT. An electronic device which has many transistors and other semiconductor components integrated into one piece of silicon.

LOGIC CELL. The generic term for a basic building block of a general-purpose logic device.

LOGIC SIMULATION. A means whereby a logic design can be evaluated on a computer before actually being built. The computer simulates the behavior of the components to predict the behavior of the overall circuit.

LOGIC SYNTHESIZER. A compiler module that uses several algorithms to minimize gate count, remove redundant logic, and utilize the device architecture as efficiently as possible.

MASK PROCESS. A series of steps used to cover or coat a semiconductor surface to conceal specific areas for selective depositing or etching.

METALIZATION. The process of connecting the various elements of an integrated circuit by placing a layer of metal over the entire wafer and then selectively etching away unwanted metal. A photolithographic mask defines the pattern of connections.

NETLIST. A data file generated by the design synthesis process that describes IC functionality as a list of circuit elements and their network of interconnections. The netlist is used by the foundry in the placement and routing process.

NON-BEHAVIORAL TESTING. A method of testing an integrated circuit to see that each logic element and interconnection performs its defined function, regardless of how the device will be used.

OBSERVABILITY. The ability to determine the value at a circuit node inside a device using its external pins. For a digital circuit the value would correspond to a logic 1 or 0.

PROCESS YIELD. The number of devices produced that test good divided by the total number of devices tested, expressed as a percentage.

PROGRAMMABLE LOGIC DEVICE. A logic device programmed at the customer site. It contains various configurations of gates and flip-flops.

QUIESCENT CURRENT. The power supply current drawn by a circuit after the inputs have changed state and the circuit is in the steady-state non-switching condition. The IEEE symbol for quiescent current flow in a CMOS device is I_{DDQ}.

QUIESCENT CURRENT TESTING. A test methodology used in the testing of CMOS devices to improve the detection of defects and failure mechanisms. The test is performed by measuring the device supply current flow I_{DD} in the quiescent logic state. This test method is referred to as I_{DDQ} testing.

SCAN CHAIN. A design for testability technique where the storage elements are connected together in a serial shift register chain. This facilitates testing because it simplifies the on and off loading of data into all the sequential elements in a design.

SCAN DESIGN. A design method in which special circuits are used to convert a sequential circuit to a combinational one to ease test generation.

SCHEMATIC ENTRY. The process of describing a circuit by entering a detailed logic diagram into a computer system.

SEQUENTIAL LOGIC. A logic circuit whose operation depends on both present input signals and previous operations or states. The logic requires memory elements for remembering past states.

STRUCTURAL TESTING. A form of testing used to verify the operation of each cell and respective interconnections internal to an IC. Structural testing provides a degree of fault coverage unattainable from functional testing only.

STUCK-AT FAULT. A physical defect that causes a circuit node in a device to remain at a fixed level. For a logic circuit, the level would correspond to a logic 1 or 0.

SYNCHRONOUS. 1. Describing a sequential logic system wherein all operations are synchronized to a common system clock. 2. Signals whose behavior and timing are synchronized to a clock.

SYNTHESIS. Translation and optimization of a hardware description language specification into a gate level implementation.

TEST ACCESS PORT. A device interface controller used to control the access and function of built-in test hardware. The interface is defined by IEEE standard 1149.1-1990.

TEST STRUCTURE. The nature and organization of the test program, including: the origin, form, and type of test data; the destination of the results; and the procedures used to control test operations and process data.

TEST VECTOR. A pattern of bits applied to a circuit in test to detect a fault. A test vector is also referred to as a test data pattern.

VERTICAL TEST INTEGRATION. A testing structure that provides for testing capability of the system, the sub-systems, the PC boards in each sub-system, and the major devices on each PC board. Testability of each device is integrated into the system test structure. An integrated hierarchical test architecture referred to as Electronic System Test Automation, and The Fourth Generation Test Methodology.

VIA. An interconnection between insulated metalization layers of an integrated circuit used to provide a conductive path between layers. Vias provide the same function as plated-through holes provide on printed circuit boards.

WAFER. A round slice of pure silicon which is used in the fabrication of integrated circuits. Many circuits can be built on one wafer. The present standard wafer diameter is 30 centimeters.

INDEX